MASTER IN COMPUTER SCIENCE

Implementation of a chatterbot agent including believable personality

Barthelemy, François; Gries, Samuel

Award date:
2004

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique.

Implementation of a chatterbot agent including believable personality

François BARTHELEMY & Samuel GRIES

Thesis presented in view to obtain a master degree in computer sciences

Academic year 2003-2004

Abstract

In our modern world, the virtual becomes more and more important. Many virtual worlds will inhabit the close future and these worlds will have a crucial role to play (as an example in the Stock Exchange domain). In such worlds, each actor is represented by an avatar. In the view of having a better embodiment of the actors in these worlds, a conversational agent (a chatterbot) based on agent's technologies has been developed. This chatterbot has been given a proper personality influencing the way it behaves. The chatterbot and some related works in the concerned domains will be exposed in this document. So, the agents (and specially the believable agents), the chatterbots and the insertion of personality in a program will be tackled.

Keywords: agents, believable agents, chatterbots, synthetic personality

Résumé

Dans le monde d'aujourd'hui, l'importance du virtuel est de plus en plus grande. De nombreux mondes virtuels existeront dans un futur proche et ces mondes auront un rôle primordial à jouer (notamment dans le domaine boursier). Dans de tels mondes, chaque acteur est représenté par un avatar. Afin de permettre une meilleure implication des acteurs dans ces mondes, un agent de conversation (chatterbot) se basant sur des technologies 'agents' a été développé. Ce chatterbot s'est vu doter d'une personnalité propre qui influence son comportement. Dans ce document sont exposés le chatterbot développé par les auteurs ainsi que des états de l'art dans les différents domaines concernés. Ainsi, les agents (et plus spécifiquement les 'believable agents'), les chatterbots et l'introduction de personnalité dans un programme seront abordés.

Mots-clés : agents, agents crédibles, chatterbots, personnalité virtuelle

Acknowledgements

The work presented in this document is based on our internship which took place during the last four months of 2003 at UTS (University of Technology, Sydney). The realisation of this document is not only the work of two students. So we would like to thank all the persons who helped us, in one way or another, to achieve this work.

We would like to thank Ms Noirhomme-Fraiture, our promoter, for following us during all the preparation of this thesis and for her good advice.

We would like to thank Mr Simoff and Mr Debenham, our internship masters, for all they did to help us enjoy our stay in Australia, as well for the work as for our pleasure.

We would like to thank Mr Berleur and Mr Schobbens for the good advice they gave us.

We would like to thank Mrs Kinet, Mrs Hermann and Mrs Binda for their useful corrections of our English.

We would like to thank Benjamin and Xavier for their friendship and collaboration during our stay in Australia.

And finally, we would like to thank our relatives, friends and girlfriend for their indulgence to our unavailability.

Content Table

Figures Table

Introduction

Nowadays, we have to face with more and more technological progresses. Indeed, there has been an enormous amount of work in the development of electronic commerce systems since the late 1990s. These systems try to represent virtual commerce environments. The challenge is to emulate the real market worlds by representing them by user-friendly virtual worlds in order to allow everyone, expert or beginner, to penetrate them. So, when we compare the number of the virtual worlds today with what they were five years ago, we can see a huge evolution. Many projects (mainly on Internet) will be operative in a close future. They will permeate the life of many people. This arrival of the virtual in the real raises a lot of different problems, as well technological as ethical. In this document, we will focus on one problem: the integration of people in these worlds. Indeed, as most people are not used to such worlds, they feel lost or 'not in the right place' when they enter them. So, we have developed one conversational agent (also named chatterbot) which can help the users to have a better embodiment in these worlds. This chatterbot will behave like a human; it will have its own personality, identity and will act and react consistently according to its mental background. This is one of the possible uses of our chatterbot, and many others can be imagined.

After this introduction, the different domains or techniques used in the achievement of our chatterbot will be presented. In the first chapter, a general introduction to the agents will be given. Several questions will be answered: what they are, what their features are, how they can be classified, etc. In the second part of this chapter, we will deal with one specific kind of agents: the believable agents. In this section, using the Oz project as main example, we will present

different features of the believable agents and will take an interest in their structure.

The second chapter will deal with the chatterbots. As the program we have done is a chatterbot, the main point developed in this section will be the structure and the different architectures available to create such programs. For this purpose, three different architectures of well-known chatterbots will be explained and then analyzed.

In the third chapter, we will focus on the insertion of personality in a program. This means that we will make a summary of what has already been done on this subject. Indeed, this section will display different ways of how to model personality. This state of the art is actually based on diverse related works such as the famous OCC model or personality model for long-term computer games as well. The chapter will end with an underlining of the different concepts mainly used in the modelling of personality.

The fourth chapter will explain our chatterbot. In this final section, we use what is detailed in the previous chapters. In the different sections, the following points will be presented: the context, the architecture (high-level and then in detail), the possible uses, the moral impact, etc. The architecture in a very low view is explained in the appendices.

Chapter 1: the agents

1.1 Introduction to the agents

1.1.0 Introduction

There isn't a general definition of what an agent is. Each paper describing a so-called agent gives its own definition that matches its own agent. In the first part of this chapter, a brief introduction to what agents are will be given. Then, in the following part, a small presentation of the general features that are given to the agents will be described. Finally, in the third part, a small classification of the different kinds of agents will be done.

1.1.1 What is an agent?

Looking in a dictionary for the definition of an agent, we can see that the word agent comes from the Latin verb *agere* that means *to act*. So we can consider an agent as everything that can act. According to that definition, every human and most of the animals are agents. But what about computer programs? A lot of people think that a software agent is just like any program. Indeed, if we consider an agent as *"anything that can perceive its environment through sensors and act upon that environment through actuators"* [Russell and Norvig, 2003] and if an environment is perceived as whatever provides input and receives output, then every program is an agent. So we have to be more restrictive upon the features that are required from an agent. Most papers agree on that point, but they disagree on the features necessary to speak of an *agent*.

For Hayes-Roth [Hayes-Roth, 1995], an agent can perceive its environment, perform an action in this environment but it also can think and have some power of reasoning to determine the action to perform.

For Wooldridge [Wooldridge and Jennings, 1995], "*an intelligent agent is one that is capable of flexible autonomous action in order to meet its design objectives, where by flexible, I mean three things:*

- *reactivity : intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives;*
- *pro-activeness : intelligent agents are able to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives;*
- *social ability : intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives*"

Wooldridge insists on the fact that the agent must know very well the environment, it can interact and react with every part of this environment. He is the first to give such importance to the social ability.

Smith, Cypher and Spohrer, with their KidSim Agent, define an agent as "*a persistent software entity dedicated to a specific purpose.*" [Smith and al., 1994] The authors introduce the persistence to distinguish the agents from some subroutines; they speak of specific purpose to avoid confusion with multifunction applications (agents are much smaller). In this case, the point is put on the features of persistency and specific purpose. The first is often considered as an important addition to the definition of agency, the latest is less important though a lot of agents definitely have a specific purpose.

The PAW Agent ("Personal Adaptive Web Agent") is an agent that records the user's internet activity, looking for the user's interests. In a second step, the agent finds on the web documents which potentially interest the user. Then he suggests these documents to the user. The developers of the PAW Agent have given their own definition of an agent: "*An intelligent agent acts on*

instruction of the user and can use his knowledge of the user's interest and wishes to do his work" [Lepuschitz, 2000]. They claim that an agent has to be autonomous, social, reactive and proactive. The autonomy means without intervention of the user, social means that it interacts with other agents, reactive means it reacts on the changes on the environment and proactive means it can take the initiative. Once again, the definition of the agent and the features it must have match what is done by the PAW Agent.

Stan Franklin and Art Graesser have analysed the different definitions of agents and then decided to write their own with the features they think to be the most important. For them, "*An **autonomous agent** is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.*" [Franklin and Graesser, 1996] If we look closer at that definition, we can see it is a generic definition not only valid for software agents but also for animals and humans. This definition also allows a thermostat to be an agent.

This definition seems to be the most natural and adapted to all the situations. Using it, we can say now whether a computer program is an agent or not. So for the purpose of this thesis, each time we will speak of an agent, it is an agent that satisfies that definition.

1.1.2 General features of an agent

But there are a lot of different classes of agents, how could they be categorized? By making a list of all the features of agents, we will be able then to categorize them into more specialized categories. At first, we have to make the list of all the different features that are the most generally given to an agent.

- Reactivity: the agent can sense its environment and do an action as a reaction to this environment on basis of what it has sensed.
- Pro-activeness: it acts on the environment without any stimulus from the environment and can take initiatives.

- Autonomy: it can act without the intervention of a human or another program.
- Persistency: it is not a temporary program; it is a continuously running process.
- Social ability: it can communicate with other agents (including people).
- Adaptability: it can change its behaviour on the basis of its history and the actions it has done. It learns from the experience.
- Flexibility: its actions are not scripted.
- Mobility: it can move itself from a machine to another.
- Character: it has a believable personality and an emotional state.

As said by Stan Franklin and Art Graesser, some features are necessary for a program that would be an agent. According to their definition, every agent must be reactive, autonomous, pro-active and persistent. The other properties are optional and there is no agent that has all of them (this is not useful). So we can distinguish the agents by the additional properties they have.

Finally, we have to talk about the level of abstraction (LoA); LoA in a computer scope is a concept that has been thought by Luciano Floridi and J.W. Sanders from the University of Oxford ([Floridi and Sanders, 2001]). They say that a program can be an agent or not, depending on the LoA where the analysis is done. But what is a level of abstraction? The best way to clarify what it is is by giving an example. So we will re-use the example given by Floridi and Sanders: three people are talking: Anne is a collector and possible buyer, Ben is a mechanic and Carole is an economist. We are able to hear what they are saying:

a) Anne observes it has an anti-theft system; it is kept garaged when not used and has a single owner.
b) Ben observes the engine is not the original one and it has been recently re-painted.
c) Carole observes the engine consumes too much, it has a stable marketable value and its spare parts are too expensive.

The participants view the object under discussion according to their own interests. They each see the object at their own level of abstraction (the other details are ignored). "*A LoA consists of a collection of observables, each with a well-defined possible set of values or outcomes [...] Each LoA makes possible an analysis of the system, the result of which is called a model. Evidently, an entity may be described at a range of LoAs and so can have a range of models.*" [Floridi and Sanders, 2001] This definition can be adapted to the agents. There are a lot of possible points from where an agent can be analyzed: the point of view from the user who uses the agent for the first time, the point of view of the frequent user or also the one of the programmer. We will explain these different LoA by giving another example.

Suppose we consider a program that plays noughts and crosses (OXO) and learns from every game it plays. The program only plays the 'O' characters. In the first games, the program plays at random choosing one of the possible moves. At the end of each game, the computer memorizes the boxes where it has played and adapts the probability of each of these boxes (increase sharply the probability if victory, increase steadily in case of a draw and decrease in case of defeat). After enough games, it becomes impossible for the random selection to produce a losing play.

Now, let us consider three different LoAs for this 'agent'. (Note that for this example, we will use the definition of an agent given by Floridi and Sanders, i.e. an agent must be autonomous, reactive and adaptive)

a) The single game LoA: the observer can only see the boxes played by the program. At this LoA, the program seems to be reactive and autonomous (it reacts to the moves of the player autonomously) but it fails to be adaptive (the player has no means to determine how the computer chooses the next move).
b) The tournament LoA: a sequence of games is observed. The program is still reactive and autonomous but this time, it is also adaptive. Indeed, if the player observes the sequence of results, he can see that the rule that determines the next move is changing over time. Thus, at this LoA, the program can be considered as an agent.

c) The system LoA: we finally observe the code of the program; it is the lowest LoA possible. At this level, the program is still reactive and autonomous but it fails to be adaptive: what seems at the second level to be an evolution of the rules to determine the next move is in fact a simple deterministic update of the program state. At this lowest LoA, the program fails to be an agent.

So, in conclusion of this presentation of the levels of abstraction, we can say a program can be analyzed from different LoA and the fact that this program is an agent or not can depend tightly from the level of the analysis.

1.1.3 Classifications of agents

There are a lot of possible ways to classify the agents. For the purpose of this thesis, we will categorize them using the same classification that Franklin and Graesser had defined in 1996. After that, we will have families of agents and so we will be able to determine the category of each agent. The classification that will be done is quite like a biological classification. It is a hierarchical decomposition where each cross is equivalent to a question and each node is a kind of agent.

At the first level, the classification is done between the biological, robotic or computational agents. On the next level, the distinction can be done between the artificial life agents and the software agents. For the decomposition on the next level, it can be between the purposes of the agents: task-specific, entertaining, computer virus.

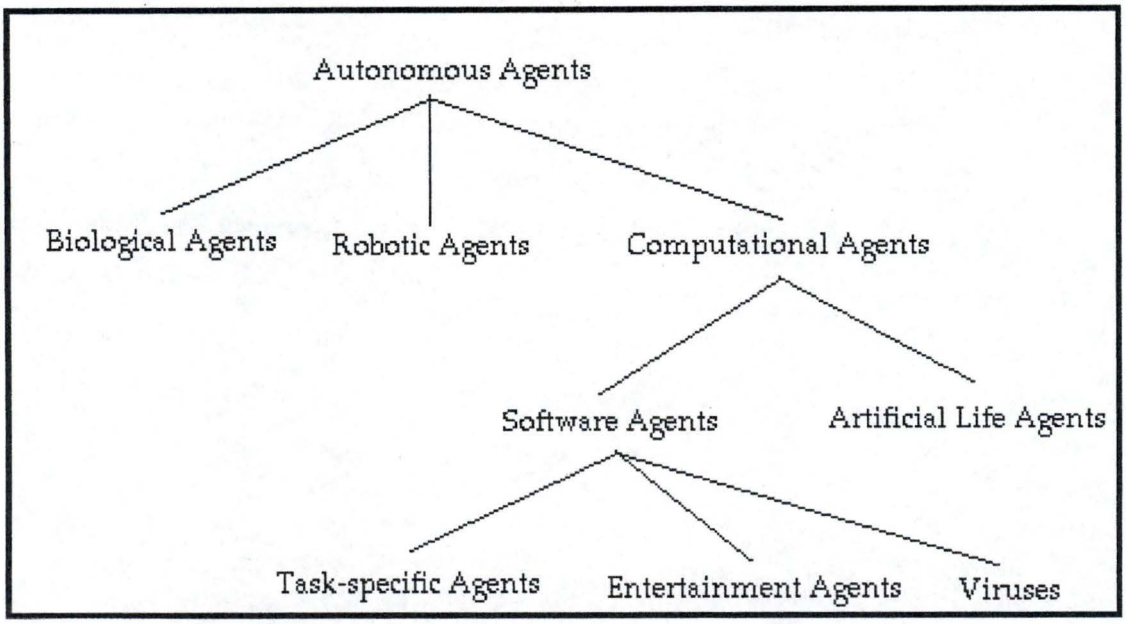


Figure 1-1: decomposition of the autonomous agents from [Franklin and Grasser, 1996]

At this point, we have a very satisfactory classification but it would be useful to push it further. Now, we can make this further classification with regards to the extra features of the agents. We can classify the agents on the fact they have a property or not. After that classification, we may speak of planning, communicative, mobile, social agents, etc. Making things easier, we may rename some of these definitions in a more convenient manner.

As an example, every agent according to the Wooldridge's definition, i.e. reactive proactive and social, could be renamed as a communicative agent. Indeed, according to our definition, reactive and proactive are features necessary to be considered as an agent. The only extra feature of the Wooldridge's agent is the social ability, so we may speak of communicative agents.

Other methods could have been used to classify the software agents. For example, the designers of such systems would prefer a classification according to the architecture of the agents. This could be a very interesting classification but, as the explanation of the different architectures to build agents is out of the context of this thesis, this classification will not be developed here.

1.1.4 Conclusion

All the software agents are programs, but not all the programs are agents. In this chapter, a definition of an agent has been proposed. Then, using the different features of the agents, we have developed a way to determine if a program is an agent or not, and a way to classify the agents into different categories.

1.2 Believable agents

1.2.0 Introduction

This section will be about one specific category of agents: the believable agents. In the first part, we will discuss about the definition of a believable agent, their purpose and the environments in which they are used. Then, in the second part, we will go over the different features that are required by a believable agent. In the next section, we will discuss about the different structures that are often used to program these agents. We will then make an analysis of the different works existing on this matter. Finally, we will close this chapter by developing our conclusion on that theme.

1.2.1 What is a believable agent?

In many virtual environments, it is often hard to feel totally embodied in the world. These virtual worlds are lack plots and characters that inhabit them. Some new environments more likely to a global immersion of the users have been developed in many projects like the OZ project. Better characters are also developed by some teams of researchers; these characters whose aim is to help the embodiment of the user in the world are called believable agents.

The problem to create these virtual worlds is to create a place where the user can feel free but also have some artist-shaped experience. This problem will not be discussed in this thesis. The other main problem is to create characters that are as rich as characters in novels while also being interactive. The problem is that traditional AI has developed tools to compute agents that solve problems but they haven't done anything that could help to program emotions or social relations. So, because creating believability is different than creating intelligence or realism, using these tools for designing believable agents will not lead to the wanted result. Another kind of tools and architecture has to be developed for the creation of believable agents. To know what kind of architecture and tools is necessary to develop powerful believable agents, we have to understand the concept of believable agent and its purpose fully.

The best people to create or imagine believable agents are artists, novel writers, etc. But these people don't know how to program an agent. At the opposite, AI researchers know how to design agents and how to program them but ignore the mechanisms that make look an agent believable. The challenge is then to combine those two disciplines to produce autonomous, interactive agents having the qualities that have made the non-interactive characters of traditional media believable. It seems, regarding to the work done on the OZ project, that the best way to combine these two disciplines is by building tools to support the artistic task.

1.2.2 Features of believable agents

The purpose of a believable agent is to cause emotions to people and to seem alive. So, to meet that aim, the believable agents don't need to be intelligent neither to be realistic but they will have strong personalities. Now, we will explain more deeply each of these three points.

The believable agents don't need to be intelligent because an agent that could know everything and that could do things efficiently would be highly unbelievable. Moreover, it will be sometimes desirable to have stupid characters.

The believable agents don't need to be realistic. It is impossible to compute a mind that would exactly be like a human mind. As said in [Reilly, 1996], "*it is better to go with the less realistic characters which meet the audience's expectations than to go with the more realistic characters which don't*". Unrealistic characters can be very believable. So, often, the aim is not to create a plausible agent but to create a good character.

Strong personalities will make the believable agents more interesting. A good way to make the agents more believable is to let the personality of the agents influence its movements, way of speaking, etc. As the classic architectures to program agents are not adequate to compute personality, it will be necessary to let the personality permeate the architecture.

1.2.3 Structure of believable agents

For the purpose of this thesis, we will rapidly introduce an architecture for building believable agents. This will be done by analysing the architecture developed by the members of the OZ project. The OZ project himself will be presented in the following section. They chose to create believable agents by using broad agent architectures. These architectures have three properties: they use a broad set of capabilities, each of them is somewhat shallow and all the capabilities are highly integrated. In the next paragraphs, each of these properties will be developed and explained.

- A broad set of capabilities: believable agents need to behave like interesting characters in a virtual environment including human users. So, because humans are highly unpredictable, they need to be capable of handling a lot of different situations. If an agent can't handle a situation

correctly, it will definitely cause the disbelief of the human users. The believable agents will need to have a lot of capabilities like emotions, memory, social skills, reactivity to the environment, language understanding, language generation and many others.

- Each capability is shallow: the users of the virtual world are like people reading a novel or people watching a movie: they want to suspend their disbelief. So, as long as the agents in the virtual world don't act out of the character, users will tend to find them believable, coming up with reasons for unusual behaviour. So, in an architectural view, it means that implementing all the capabilities superficially will be sufficient. The capabilities can be computed deeply but it isn't often necessary in a context of believable agents. This kind of architecture will be, in this situation, better than the classic narrow-and-deep methodology (which is corresponding to have only a couple of capability which are developed).

- The capabilities are integrated: in this kind of broad-and-shallow architecture, an integration of all the capabilities will lead to powerful results. Indeed, a large set of capabilities become much more powerful when they are tightly integrated. For example, reinforcing the emotions of an agent by expressing these emotions by movements of its face will have great results to the belief of the human users. There are a lot of other possible interactions between the different capabilities (emotions and natural language, social skills and emotions, etc.)

To implement this broad-and-shallow methodology, the members of the OZ project have designed and built the *Tok Agent Architecture.*

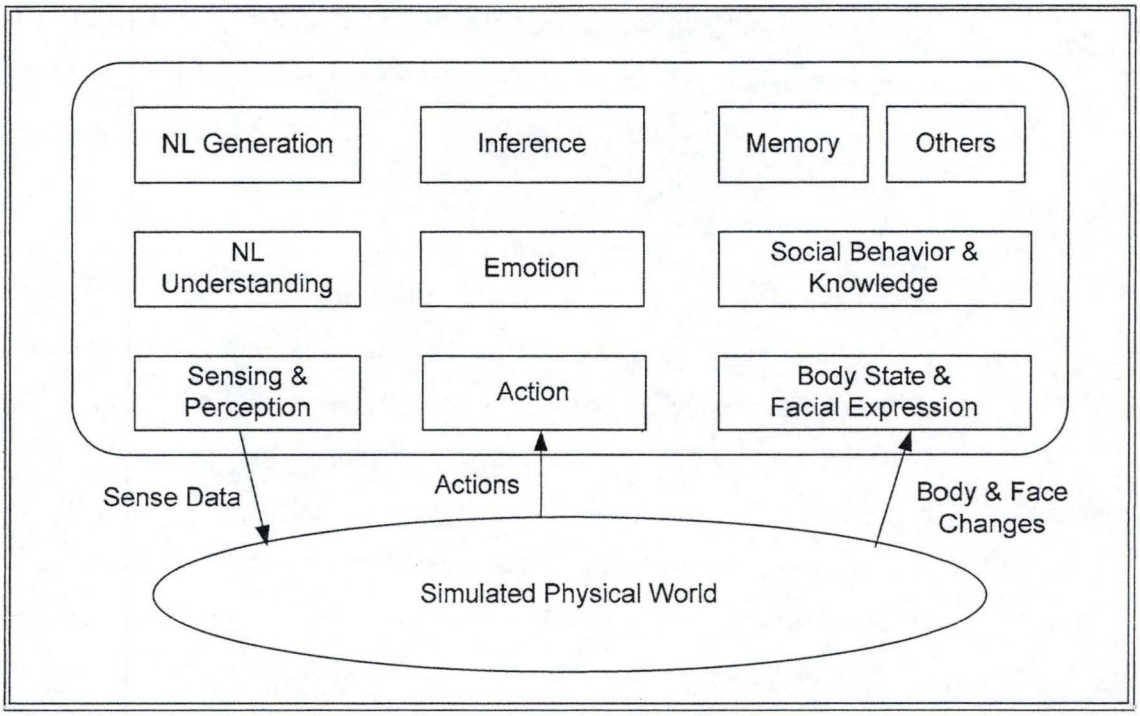


Figure 1-2: high level schema of this Tok architecture

Looking at this architecture (figure 1-2), we can see a lot of different capabilities. The integration of all these components is done by writing the code of all these various sub-systems in a common behaviour-based programming language: **Hap**. It is considered by the members of the Oz project as a language but as well as the architecture build using this language. Hap is the architecture for the minds of computer agents in Oz and is the reactive component of Tok. It has been developed specifically to support believable agents. The goal of the Hap architecture is to allow robust, reactive behaviour as provided by existing hard-wired reactive architectures while capturing a richer range of high-level actions. Largely inspired from [Loyall and Bates, 1991], Hap's specificities will be rapidly described in the next paragraph.

- Hap always keeps all the behaviours running together (at the same time). Each behaviour tries to perform some actions. These actions can contain other sets of actions or lead to other behaviours. For example, an agent, which has to simulate the behaviour of a student, can have different goals: eat when hungry, do sport when he wants to have fun, go out when he

wants to meet people or study otherwise. The 'going out' behaviour can lead to other behaviours like calling a friend and then going to a party. Both of these steps are subgoals that can lead to other behaviours. The 'call a friend' behaviour has a mental step that chooses the friend to call.

- All the behaviours are pre-coded; Hap does no planning in the traditional sense. Hap doesn't create any way of doing the things, it is written in its memory. When the agent has to decide which friend he wants to call, there is a pre-coded rule in his mind to determine the friend to call.

- Goal success is not necessarily a testable property of the world. This means that sometimes it is impossible to represent the state in which a goal has been achieved. In the 'study' behaviour, the agent has a goal that consists of reading a book. In the initial state as in the final state, the agent is in front of a closed book. What is important is this case is the action that has been accomplished not a particular state of the world.

- Hap supports the creation of reactive behaviours. New goals can be created (the goal to eat is created when the agent is hungry), goals can be paused then resumed (the goal 'study' can be interrupted by the ring of the phone than resumed after the communication), behaviours can be rejected when the situation changes (the goal 'study' can be rejected after the call of a friend which announces a great party).

- Hap allows multiple threads of programming. This allows multiple behaviours to be running at the same time.

- Hap is a general programming language; so many types of behaviours can be written using it.

1.2.4 Related Works

In this section will be shortly presented some works about believable agents. First, the famous Oz project will be exposed then another interesting work done by the University of Stanford will be introduced.

The OZ project:

Among all the projects having their theme about believable agents, the OZ project ([Bates, 1994], [Reilly, 1996]) is the most famous one. It has been developed by the Carnegie Mellon University. Oz is intended to provide a human user with the experience of living in a dramatically interesting simulated world which includes simulated intelligent and emotional agents. The virtual world they developed provides an object oriented simulation in which characters (both human and computer) can interact. The world contains many of the difficulties of the real world: it is complex, unpredictable and provides incomplete information.

The physical world contains and models all the physical aspects of the system including the bodies of the character. As shown in the figure 1.3, these physical bodies are connected to their corresponding minds through perception/action channels. When a character wishes to perform an action, it sends an act frame to its body in the world. The characters perceive the world through *sense data*. The sense data are either active or passive in the OZ world. They are active when they have a sufficient intensity to be sensed by the agent. They are passive when the act of sensing requires an action from the agent; so passive data are acquired by querying the world.

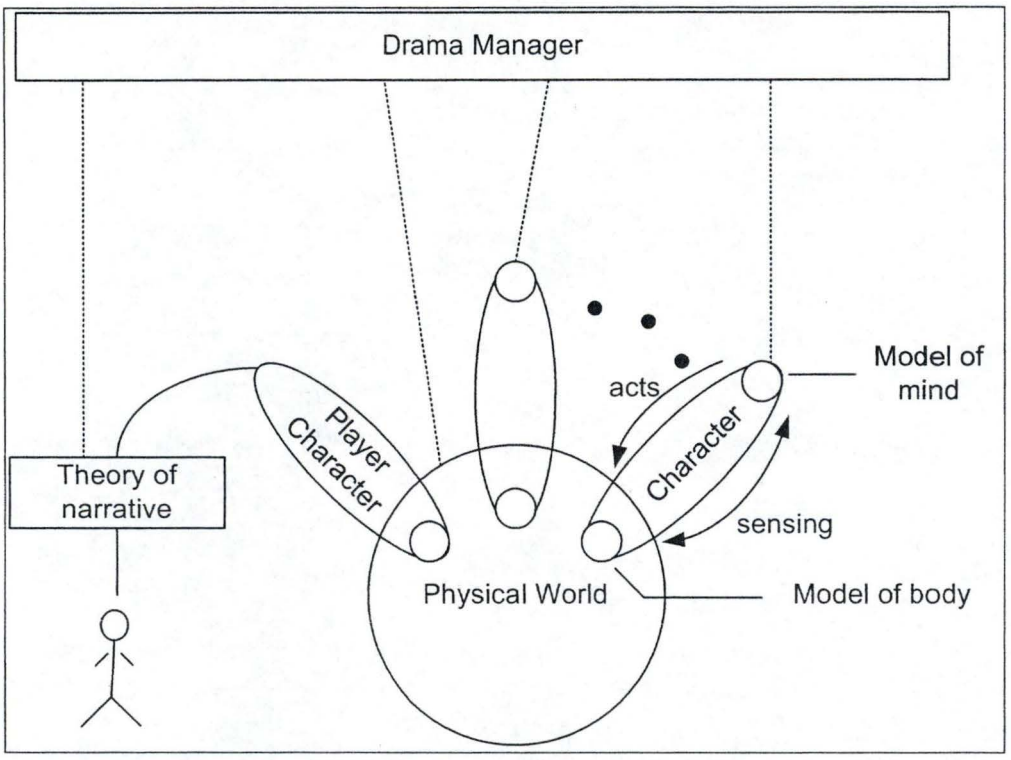


Figure 1-3: general structure of the OZ world

One idea proposed by Reilly is that emotions and social behaviour are two excellent ways making the agents more believable. The social factor is the origin of an important set of emotions. Emotions like love, anger, jealousy, envy, etc are only possible when the agent has a social behaviour. Some emotions can only be expressed in a relationship. In another way, the relationships have a big influence on the emotions. (What a friend says can have a big influence on the happiness or other emotions). Another important point is that the integration of the social behaviour and the emotions make agents more believable. This leads to richer relationships that can change over time.

In order to compute the emotions in the mind of the agents, the members of the Oz Project have developed one specific tool *Em*. This tool will be developed in the chapter 3 of this thesis.

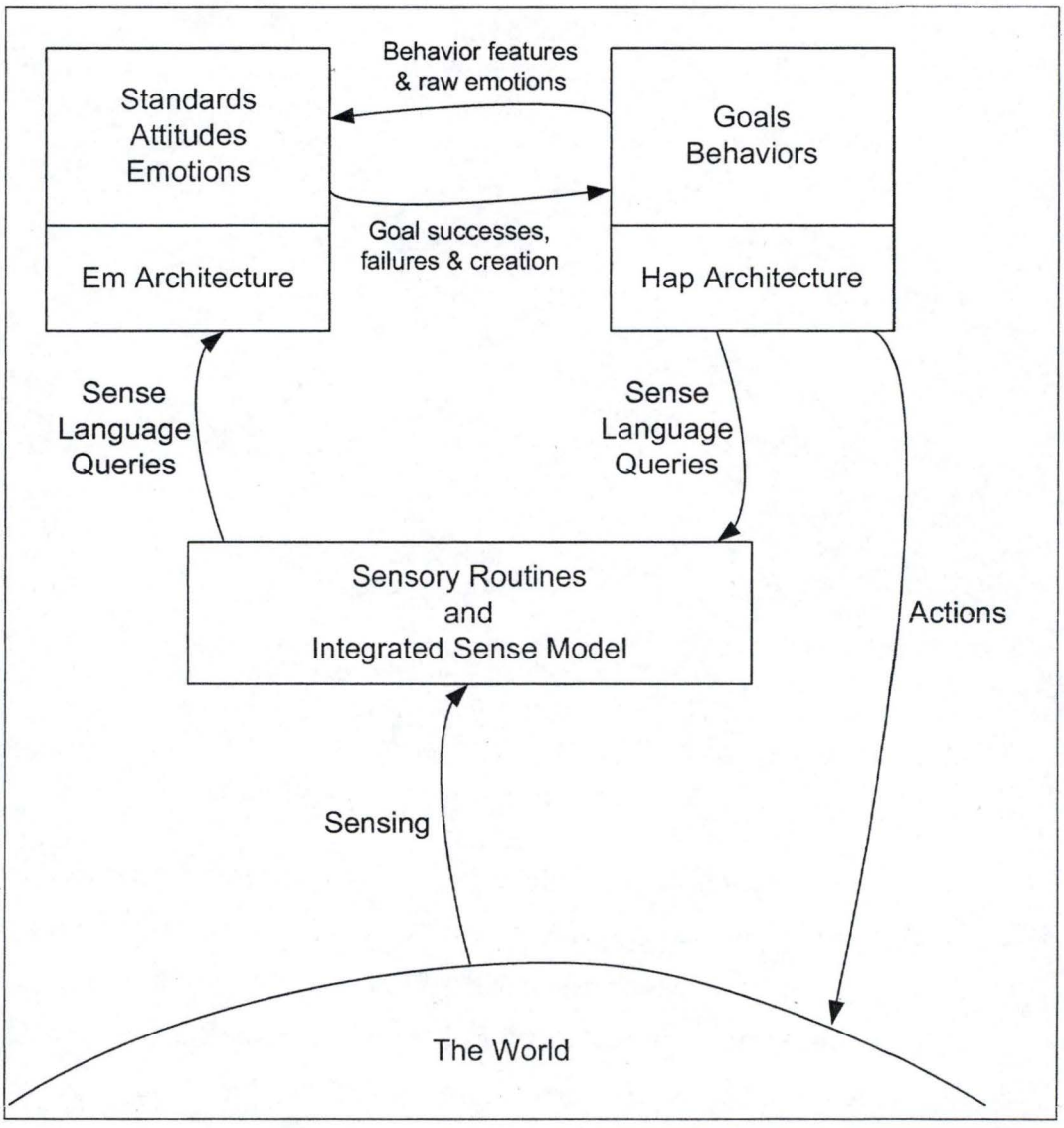


Figure 1-4: global architecture of the OZ world

The Virtual Theater Project

The University of Stanford has developed a project to "*provide a multimedia environment in which users can play all of the creative roles associated with producing and performing plays and stories in an improvisational theatre company*" ([Doyle, 2001]). In this environment, intelligent agents play the roles that aren't assumed by the user. These agents that play actors bring life-like qualities to their performance; they don't only follow scripts. They also

improvise; they may react every time differently to a same 'stimulus'. This takes part in the creative process.

The current research focuses on building individual characters which can take directions from the user or the environment, and act according to these directions in ways that are consistent with their unique emotions, moods, and personalities. As part of the researches, the team of the project have implemented several systems for exploring how agents interact and how the users can guide them. They have developed very different worlds: "*Animated Puppets*", "*CyberCafé*", "*The Forest Sauvage*", "*Tigrito*" and "*Master/Servant Scenarios*". Among all these projects, we will focus on the CyberCafé.

Three roles are to provide in the CyberCafé simulation. The situation takes place in a bar, they are two customers and one waiter; the waiter and one customer are played by autonomous agents, the remaining customer is directed by the user. Each of the actors has its own personality. To implement it, three traits have been chosen to define the character's personality: activity, friendliness and self-confidence. Sometimes, the value of a trait can depend on other components. As an example, a choleric character who is usually calm can become hostile after a certain threshold for anger has been reached. For the purpose of this experiment, ten experimental personalities have been defined.

They made a lot of tests with different configurations of the agents and so, they showed by their experiments that "*synthetic actors behave with respect to their personality by performing actions showing relevant characteristics most of the time. [...] they (the users) can recognize a character's personality, and they find the characters largely believable and interesting.*" ([Hayes-Roth and Rousseau, 1997])

1.2.5 Conclusion

Amongst the chapter, a general introduction of the concept of believable agents has been done. After giving a definition, the problem of the non ability of the normal programmers to program such agents has been introduced. We have tackled this problem by giving the sources of the believability of an agent. Indeed, a list of features that make the agent more believable has been developed. Moreover, an explanation of the general structure of these agents has been achieved. Some parts of this structure will be developed more deeply in the chapter 3 which focuses on the introduction of personality in a program. Finally, two main works in the domain of the believable agents have been introduced.

Using what has been summarized in this chapter, a normal programmer will now be able to approach the problem of developing a believable agent with a better conduct than before.

Chapter 2: The Chatterbots

2.0 Introduction

When we compare today's world to the world of our fathers, and if we have a look at the different forecasts of what the future life could be, we can say the world is changing very quickly. What was science fiction a few years ago is sometimes present now or close future. We can see in today's world that the computers are more and more present; the virtual world is entering the real one. In such a world, one point is the object of a lot of work: the domain of the interactions between humans and the machines. The part of this work about the conversation is called chatterbots. There are programs that help a human user to embody in the virtual world by giving the human the illusion that he is speaking with another human.

In this chapter, we will deal with the chatterbots. In the first part, we will give a definition of what they are, then their history and some of their context will be developed. In a second part, a large description of some structures of chatterbots will be explained. Then, according to their features, we will propose a classification for the chatterbots to finally give our conclusion about this theme.

2.1 What is a chatterbot?

According to [1], a chatterbot is a program that attempts to simulate typed conversation, with the aim of at least temporarily fooling a human into thinking they were talking to another human. While it is true that a good understanding of a conversation is required to carry on a meaningful dialog, most chatterbot do not

attempt this. Instead, they attempt to pick up cue words or phrases from a person which will allow the bot to use pre-prepared or pre-calculated answers; these one can move the conversation on in an apparently meaningful way without requiring them to know what they are talking about. The classic early chatterbots are ELIZA, PARRY and SHRDLU. More recent programs are Racter and A.L.I.C.E. In 1994, the term 'chatterbot' was established in the AI lexicon by Michael Mauldin of Carnegie Mellon University, in his account of entering the Loebner contest.

2.2 History and context of Chatterbots

2.2.1 ELIZA

During the 1960's Joseph Weizenbaum created ELIZA. ELIZA was the first ever chatterbot in the history. 'She' created a storm of public interest in AI, as it helped thousands overcome their personal problems. ELIZA was a rogerian psychiatrist, one that takes the user's statements and rephrases them in a question (example: User says "My mother's dog died recently" than ELIZA says "Tell me more about your mother"). Though sometimes they may have seemed ambiguous, people actually felt ELIZA could take care of their needs just as well as any other therapist. They became emotionally involved with ELIZA, even Weizenbaum's secretary demanded to be left alone with the program.

When people had started calling ELIZA intelligent, Joseph Weizenbaum went into an uproar. Technically, ELIZA was actually unable to understand people's personal problems to the depth of any other human being. ELIZA could only manipulate syntax (grammar), and check for key words. Certainly, if someone had no knowledge of ELIZA being a program, one could easily conclude that it behaved like a human conversing, although it never really necessary understood everything to the detail that humans do.

2.2.2 Turing Test

In 1950, Dr. Alan Turing, a British mathematician who is now considered the "Father of AI" proposed the "Turing Test" for intelligence. Simply put, the Turing Test boils down to the question: "Can this machine convince the human to think that it's human?". Specifically, the machine is a natural language system that converses with human subjects. In the Turing Test, a human (the judge) is placed in one room, and the machine/or another human is placed in another room. The judge may ask questions or answer questions posed by the computer or another human. All communication is done through a terminal, input is done by typing. The judge is not aware whether or not the subject that he/she is talking to is either a human or a computer before the conversation begins. Supposing that the judge was conversing with a computer, during and after the conversation, he/she must be "fooled" into thinking that the machine is a human in order for the machine to pass the Turing Test. There are actually very many pitfalls to the Turing Test, and it is in fact, not very widely accepted as a test for true intelligence.

2.2.3 Loebner Contest

Today, the Loebner Contest is an instantiation, a modern version of the Turing Test. The criticism surrounding the Loebner contest deals with how the Turing Test is carried out. The goal of the contestant is to fool or trick the judge into thinking that his program is a human. Such a prospect does not encourage the advancement of AI.

For example, messages are transmitted via text, as the subject (human or computer) types, the judge sees the text that is being typed, live. Thus, many contestants have been forced to emulate typing conditions of humans, i.e. text that is output comes out at varied speeds, sometimes words must be misspelled and corrected, incorrect punctuation is often used etc. Even then, the programs in the contest usually talk about only one subject (to talk about everything present in our culture is simply impossible, at least for a natural language system that

understands only words, syntax and semantics and not really what they look like, what some objects really do etc.). If the judge picks another subject to discuss, the programs usually try to divert the attention of the judge. Programs have even tried to use vulgarity or an element of surprise, to get the judge excited.

2.3 Structure of chatterbots

In this chapter, we will present some tools and architectures that are used to develop chatterbots. First, we will expose how Alice (probably the most famous chatterbot for now) decides which sentence to say and how these sentences are stored. Then, we will show another kind of chatterbot that learns everything from the experience. Finally, we will have a look at JFRED which is a chatterbot that has a structure of agent and so we will see how the different parts of chatterbot can be organized.

2.3.1 Alice

In the 1990's, Dr. Richard Wallace developed a chatterbot system that could be written in an XML specification called AIML, short for Artificial Intelligence Mark-up Language, and "Alice" was born. Today, Alice and her many derivatives, or "clones", permeate the web as artificial site greeters, sales representatives, celebrities like Elvis or The Beatles, and as a novelty item on the movie web site for "AI - Artificial Intelligence". Alice runs similar to Eliza, with more tricks and a bigger brain this time, and is a very popular chatterbot in the AI community today. Probably the biggest factor of success for Alice is the fact that she is open source, drawing on many resources around the world to contribute to her further success. So, because of its simplicity and the fact it is open source, there are many people that have tried to create their own chatterbot using the structure of Alice. Another big note is the fact that Alice won the Loebner Contest, mentioned above, for two years in a row when it was written. There are around 25000 templates in her brain, and growing. Dr. Wallace's unique one liners as responses is what gives Alice her unique personality. Now, we will see

more deeply how this chatterbot is running. The particularity of this chatterbot is his AIML language. We will describe it in the following paragraphs.

AIML is an XML-compliant language that is easy to learn, and makes it possible for everyone to begin customizing an Alicebot or creating one from scratch within minutes.

The most important units of AIML are:

- <aiml>: the tag that begins and ends an AIML document
- <category>: the tag that marks a "unit of knowledge" in an Alicebot's knowledge base
- <pattern>: used to contain a simple pattern that matches what a user may say or type to an Alicebot
- <template>: contains the response to a user input
- <srai>: redirects the generation of answers to another category

There are also 20 or so additional more tags often found in AIML files, and it is possible to create our own so-called "custom predicates".

The ALICE AIML includes a knowledge base of approximately 41,000 categories. There is an example of one of them in the following figure.

```
<category>
  <pattern>WHAT ARE YOU</pattern>
  <template>
    <think><set name="topic">Me</set></think>
    I am the latest result in artificial intelligence,
    which can reproduce the capabilities of the human brain
    with greater speed and accuracy.
  </template>
</category>
```

Figure 2-1: example of an ALICE AIML

The opening and closing <aiml> tags are not shown here, because this is an excerpt from the middle of a document

Everything between <category> and </category> is a category. A category can have one pattern and one template. The pattern shown will match only the exact phrase "what are you" (capitalization is ignored). This category may also be invoked by another category, using the <srai> tag (not shown) and the principle of reductionism (i.e. decrease of the number of sentences in the templates thanks to the redirections). In any case, if this category is called, it will produce the response "I am the latest result in artificial intelligence..." shown above. In addition, it will do something else interesting. Using the <think> tag, which causes Alicebot to perform whatever it contains but hide the result from the user, the Alicebot engine will set the "topic" in its memory to "Me". This allows any categories elsewhere with an explicit "topic" value of "ME" to match better than categories with the same patterns that are not given an explicit topic. This illustrates one mechanism whereby a botmaster can exercise precise control over a conversational flow.

One of the defaults of this architecture is that every time the user asks the same question, the chatterbot answers with the same sentence.

2.3.2 Leo

Leo is another kind of chatterbot, it is a learning chatterbot and, according to his author (Alan J. Brown), "by far the most advanced learning bot on the web". Leo does not have a complicated set of rules dictating what to say when certain predefined key words are triggered like many chatterbots. Leo learns from the users who converse with Leo on their own computer. It remembers every sentence that has been said by the users and works by association to answer sentences.

When someone first chats with Leo (which has a clear database), Leo only repeats what the user says. After a while, it will start to use sentences from further back in the database. So, on the one hand, starting from a clear database

can be very frustrating but, on the other hand, as Leo works by associations, it can learn any language. Once the database contains enough data, it can be very nice to chat with Leo. Bots like Leo are interesting bots because they can learn new sequences from its users and constantly expand their capabilities.

Let us now have a closer look at the way Leo works. The program has two different functions: storing information, using the stored information to edit a sentence. So, on the one hand, it saves in an Access database every sentence said by the user. It decomposes this sentence in its different words in order to store it. This way of storing is one that allows easy requests on the database to check if a word is known by Leo or not. On the other hand, in order to decide which sentence to say to the user, Leo uses very simple rules. It analyses the sentence said by the user, searching in it either the word it knows the less (the less present word in the database) or either the longest word of the sentence. This is a quite intelligent way of choosing the topic of the answer; indeed, the articles (which never have to be a topic) are in every language short and frequent words. Choosing the rarest word of the sentence or the longest will help the user see the link between the sentence and the answer. Of course, there are also other rules that influence the sentence said by Leo.

Leo won the Silver Medal in the Best Learning Bot category in the Chatterbot Challenge, an annual chatterbot contest. In the next figure, there are few quotes from a conversation with Leo.

```
You: what are you saying?
Leo: What about the weather.
You: the weather is cloudy and it is raining
Leo: Raining.
You: yes
Leo: In a way, yes.
You: do you like cats
Leo: Who couldnt like you.
You: do you like cats or dogs better
Leo: What do you like cats or dogs better.
```

```
You: I like cats better
Leo: Do you like cats or dogs better.
You: cats
Leo: Do you like the name leo.
You: yes it is a nice name
Leo: Buffy is a nice name.
You: you must really like Buffy
Leo: Then you must be happy.
You: I am usually happy
Leo: Obviously I am usually happy.
You: are you happy?
Leo: You are groovy.
```

Figure 2-2: dialog between Leo and an user

2.3.3 JFRED

JFRED is an application that can have a lot of possible uses. It has been developed in such a way that it can provide first-tier customer support via the Internet for firms that don't have enough money to pay a customer service staff. Indeed, experience has shown that some customers want more than a website, they want to chat. Another advantage of this system is that the "busy signal" that happened on a call line, or the failures because some server is down, don't occur anymore.

So, these needs from the users can be served by a kind of intelligent agent, acting as a natural language interface for a local search engine. Moreover, it can also have some kind of personality and humour. After the conversation with the user, the agent can log the conversations, which can be reviewed by a human later, as a second-tier of customer relations. According to [Nathan and Garner, 1997], there are many methods to develop such an agent, here are some:

- *Automated response to incoming email*
- *HTML forms, with the agent running via Common Gateway Interface*

- *A Java applet encapsulating a subset of a particular agent*
- *As a player within a variant of LambdaMOO*[1]
- *A multi-threaded Java application running on a TCP port*

In the case of JFRED, the choice was a Java application that implements a server which listens on a specific TCP port. Java was chosen because of its ability to manipulate network connections, its ease of program development and its portability. Another advantage of Java is that the object inheritance permits to swap easily some classes, for example the one that determines the use of grammar. That mechanism allows the agent to work with different natural languages without restructuring the system. Moreover, using Java allows the program to receive cookies and, so, using international "whois" services to decode TCP addresses and these cookies, the agent can guess some information about the user. This information can be very useful to direct the conversation or to make some statistics after all.

Let us have a look at the execution of the JFRED algorithm. It is a very simple algorithm:

```
Create an instance of the grammar object
Load a list of rules files
Listen on the TCP port

Per each TCP connection:
      Fork off a new execution thread
      Initialize rule set
      Accept "cookie" via protocol
      Load frame based on "cookie"
      Transact with interlocutor

      Per each input phrase read:
            Fuzzy logic selects rule to best describe input
            Save variables into frame, if any can be parsed
            Remap verb tenses and prepositions, if needed
            Return the response
```

Figure 2-3: pseudo algorithm of JFRED

[1] LambdaMoo is one of the oldest continuously-operating MUDs, a class of online worlds with roots in text-based multiplayer role-playing games. LambdaMoo has the specificity to use an object-oriented programming language to manipulate objects in the virtual world.

This algorithm has described the general way the program works. In this program, there are some components that are really complex. The natural language processing, which is an essential task, is one of these complex components. This natural language component itself employs fuzzy logic rules to match input phrases with candidate responses. Sets of rules, specified in simple text files, determine the behaviour of an agent, apart from its use of grammar. These rules map keyword counts and regular expressions found within an input phrase into fuzzy membership sets used to describe the semantics of the phrase. Some rules extract data from the conversation, e.g. "What is your name?", others invoke regionally-based insults depending on conversational tone, while other rules resolve fuzzy membership sets into ordered lists of candidate responses, interrogations, and suggested URLs.

So, shortly, this process includes:

- *Determining fuzzy sets of keywords*
- *Parsing grammatical constructs (regular expressions)*
- *Frame-based knowledge of country, currency, language*
- *Extraction of data from the conversation*
- *Internal state for tone of the conversation*

This system using very simple methods to store the rules allows non-technical staff to author additions to the agent; for example, in a case of a special event, an employee could create additional rules.

In the current context, they are more and more agents exhibiting personality on the Web. Now, let us have a look at an example of conversation between the agent Barry DeFacto, based on one of the latest JFRED software, and a normal user.

```
Name: Ryan
Country: United States
Language: English
Currency: Dollar
City: San Diego
Province: CA
Postcode: 92101
Provider: Electricity, Inc.
```

Figure 2-4: example of persistent learned data (stored in frame format)

```
Human> HI

Barry> what do you want?

Human> what's your name?

Barry> I'm Barry.

Human> How old are you?

Barry> I am 33 earth years old.

Human> my name is Ryan

Barry> Yes, I think that your name is Ryan.

Human> your right

Barry> Are you seeing any bots besides me? Is that how
they say it in San Diego?

Human> well, I'm looking for some. But your the best so
far
```

Figure 2-5: example between Barry and an user

We can see that Barry uses some information that he has inferred from an international database. The "name" variable is learned, and then repeated in the conversation.

Another result that has been discovered by the creators of JFRED is that the graphic representation of the chatterbot has a quite big influence on the

judgement made by the users. For example, two chatterbots accessing the same database, but one of them with a better graphical representation that the other; the users will find the chatterbot with the graphical representation more intelligent than the other.

Remark: often, people do not accept the human-like quality of an agent or a chatterbot as a measure of intelligence. This is opposed to the Turing's original criteria. Another remark is that, often the users don't find the chatterbots intelligent because they want to test their human-like quality and so ask non-natural questions to the chatterbot. They want to test if the 'person' they talk with is a program or a human. Once they talk using normal sentences to the program, the sentences produced by the chatterbot seem totally natural and often humoristic. The normal persons that speak with the chatterbots are of different natures:

- *some administer of a Turing Test by trying to determine if the agent is actually a computer program*
- *others use it as a medium to vent their ideas about how AI's should behave, as if the agent really cared*
- *some actually believe they are having "chat" with a real person*
- *a surprising number of people try to get the agent to engage in sex with them*

One interesting consequence has been the use of the Barry DeFacto agent as a front-end for a search engine. In conversation, people employ the same nouns that they would use for a search query. The fuzzy-logic rules operating on a conversational stream provides a very efficient means of cataloguing a large Web site. The result appears more organized than a keyword search (e.g. Lycos or AltaVista style search engines) and much less labour-intensive than maintaining an ontology (e.g. the Yahoo search engine).

After observing interactions with several on-line FRED's, it is apparent that the personality of the bot is essential in keeping the user's interest and drawing them back into the conversation. Even the choice of colours/graphics can affect the human's perception of the bot. The team that created JFRED experimented with combining the talents of writers, artists and musicians (along

with the required programmers) to evoke more empathy toward a constructed "virtual personality". A science fiction writer, Don Webb, was invited to develop a "history" for the agent, one based on many references to pop culture, and which could then be reference within the agent's conversation. The result combines a background narrative with music (MIDI files) and graphic design to create an aesthetic for the "robot" personality. They are now working to incorporate a speech synthesizer into the generated stream of responses.

2.4 Classification of chatterbots

Now that we have seen the main different architectures for create a chatterbot, let us have a look at their features then we will propose a classification for the chatterbots. The chatterbots can have a lot of different features. In the next paragraphs, we will describe the most important ones.

- Learning: some of the chatterbots learn from the experience. They keep in their mind what the users have told them and re-use these sentences later in the conversation or in other conversations.

- Scripts: some chatterbots have their sentences already prepared and they just find keywords in the sentences from the user to decide which sentence they have to say. Other programs re-use what the users have said.

- Agency: some chatterbots are in fact agents, this means their architecture is quite more complex and they have also other interesting properties.

- Purpose: the chatterbots can have very different purposes. Some of them exist just for the fun. Others are purely scientific: they want to prove via the Turing test that machines can think. Others have useful purposes, they give the user some information, they can help their users, etc.

- Personality: some programmers try to give their chatterbots some personality. By doing this, they want to improve the embodiment of the users in the virtual world in which they talk with the bot. They want to make their chatterbots more human by giving them one of the main features of the human nature: a personality.

- Humour: some chatterbots can be very funny, it is quite funnier to chat with someone that can make humour in the real world, so the developers have often insert a touch of humour in the chatterbots. This make the chatterbots kinder and this help to have a better embodiment of the users in the virtual world.

These features can be clustered in three groups: architecture, human characteristics and aim. In the architectural group, we can regroup the agency, the scripts and the learning ability; in the human characteristics group, there will be the humour and personality features; finally the purpose of the bot will be apart in the aim group. Each group can be used to classify a chatterbot using one point of view (one level of abstraction).

If we consider the architectural group, we can classify the chatterbots in eight different groups upon the characteristics they have or don't have.

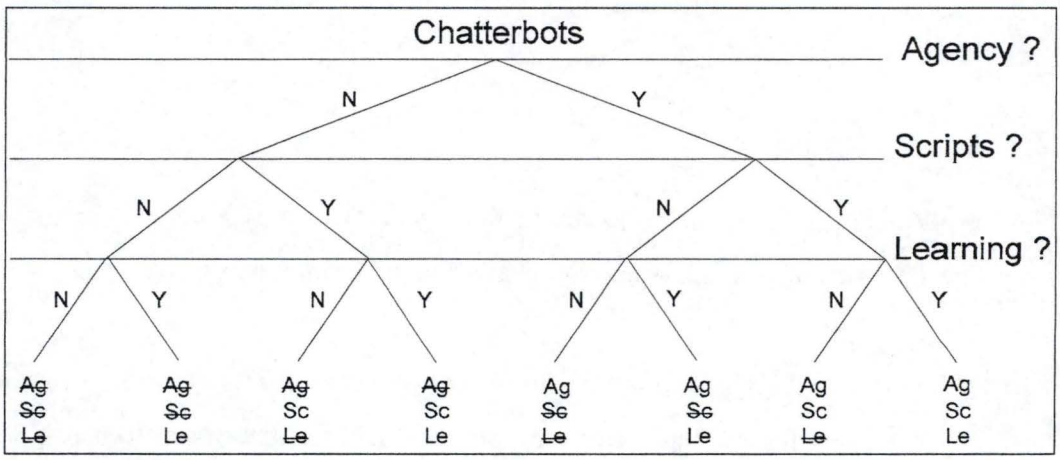


Figure 2-6: Classification of a chatterbot according to the architecture

According to this classification, Eliza belongs to the third category (~~Ag~~, Sc, ~~Le~~), Leo belongs to the second category (~~Ag~~, ~~Se~~, Le) and JFRED belongs to the seventh category (Ag, Sc, ~~Le~~). Even with those well-defined properties that have to be checked, it is sometimes difficult to say if a chatterbot has or hasn't a property. For example, JFRED keeps in its memory some details about the person he is chatting with; this property could be interpreted, by some persons, as learning, others won't say that.

Let us now consider a classification of the chatterbots upon the human characteristics they have. This classification is done using the fact a chatterbot have the features that make them more human as humour, personality (or a lot of other possible features). The chatterbots can be classified on a scale according to their human likeness. At the lowest level could be the programs that show no humour, neither personality, at the middle level could be the chatterbots that show humour or personality but not both of them and at the highest level could be the chatterbots that have the two features. According to this classification, we could find Leo at the lowest level, Eliza at the middle level (showing humour) and JFRED at the higher level.

Finally, the chatterbots can be classified upon the purpose they have. There can be a lot of different aims for the chatterbot: information, understanding of the natural language, work on the personalities, amusement, help for a better embodiment in virtual worlds (video games), etc. This third classification consists of grouping the chatterbots having quite the same aim.

Three different classifications have been proposed, each of them has its utility. Indeed, the people interested in the search for a chatterbot for a specific purpose will use the third classification; the programmers wanting to design their own chatterbot will be more interested in the architectural classification; virtual world designers searching for believability in their worlds will of course be interested in the second. The three classifications have each their own "audience".

2.5 Conclusion

Although we have seen many advances in technology, especially with AI, we still do not have software that unequivocally possesses the ability to produce conversation at a human's intelligence level. Chatterbots have utilized very sophisticate methods (as neural nets or language disambiguation parsing) as well as the more classic methods of ELIZA like Case Logic, blackboards and Natural Language Processing (NLP). Nobody and nothing can purport to have a machine that actually THINKS like a human today. At any rate, we still do not possess software that can pass the Turing Test, or display strong, genuine artificial intelligence in natural language conversation. Better chatterbots and chatterbot systems are however produced and in special contexts, they can already behave like humans.

Chapter 3: Insertion of personality in a program

3.0 Introduction:

The purpose of this chapter is to talk about the insertion of personality in a program.

The program we are going to describe hereafter is the concept known as "believable agents" or "synthetic characters". The main purpose of this kind of agent has to be as believable as possible. That is why we are going to create and introduce a personality in it. As seen before and just in order to remind it, believable agents are computer systems (also known as "lifelike computer characters", "synthetic agents", "virtual actors", etc.) designed with the purpose of provoking in those who interact with them the attribution of human-like features, such as desires, beliefs, emotions, and attitudes [2].

The study of believable agents implies the possible cross-fertilization between AI and traditional media. Artists working in drama, cinema, and literature have a great deal of experience in making believable characters, and some AI researchers want to emulate artists in developing their own computer-based characters. From this point of view, AI techniques can be seen as "expressive tools" used by the agent designer carefully (and possibly customizes) according to his own artistic goals. These goals are communicative by nature: the "AI artist" primarily wants to convey a message to an audience through his agents.

A good reason for creating believable agents is that the computer-based entertainment economy is growing very fast, and there is a strong demand for good-quality interactive characters in computer games, virtual reality, and

Multi-User Dungeons. Computer games are already populated by characters whose interactivity is however very limited. Believable agents can contribute to increase the level of interactivity and socio-emotional engagement produced by such games, or to create new forms of interactive entertainment [2].

Moreover and as quoted in the book *Creating Personalities for Synthetic Actors* [Trappl, 1997] :

Progress in computer animation has attained such a speed that computer-generated human faces and figures will soon be indistinguishable from those of real humans. The potential, such as for scripted films or real-time interaction with users is enormous. However, in order to fully realize this promise, these faces and figures have to gain autonomy, to be guided by autonomous "personality agents". But what is the current state of the art in this far less visible domain of research?

Thus, in order to understand, let us first consider how we can represent personality, emotion (both are human features) in a synthetic character in order to give the illusion of life. In fact, there are a lot of possible representations and what we are going to do now is not create a new way of representing personality but rather propose a summary of what has already been done on this topic.

The following chapter is structured as follows. First, in section 3.1, it is worth taking into account the role played by the artist and the emotions which both seem to be useful to design a personality model. Then section 3.2 refers to other works using personality, emotions, or interpersonal relationships to make an agent believable in an educational or entertainment environment. In section 3.3, we finalise this summary in order to introduce our personality model for the next chapter.

3.1 The role played by artists and emotion:

In fact, it is the artists who come closest to understand and perhaps capturing the essence of humanity that people like AI researcher seek [Bates, 1994].

As said in [Reilly, 1996], the artists are the best in knowing how to create believable characters and how to imbue them with emotions, so it is fitting to turn to the arts for guidance in building interactive believable characters.

The first contribution of artists is that they identify emotion as an important problem for building believable agents. Artists also provide ideas about how to create effective emotions for characters. These ideas are not formal, so they cannot be directly implemented, but they can help to make a number of important design decisions.

3.1.1 Emotions

One important idea about how to create effective emotional agents is that the *emotions should be specific to the character*. In other words, each character needs to be unique and its emotions need to fit its particular personality. Here are some excerpts from *The Illusion of Life* [Thomas 1981]:

These characters showed hatred and scorn in their own way, but in a convincing manner. They were equally entertaining, but they were in no way interchangeable, which points up the importance of the storyman's knowing his characters.... (p. 483)

... [I]t is the animator who must think deeply into the personality of the cartoon actors. Each must be handled differently, because each will express his emotions in his own way. (p. 487)

These quotations refer to the expression of emotions, but it is also important for the characters to have individual emotional reactions to situations as well. For instance, in Disney's *Snow White*, each of the seven dwarves might feel very differently about a single event because of his distinctive personality. And, as the quotations above state, even when characters have similar responses, they need to express those reactions individually.

Another important idea from the arts is that the *characters' emotions need to be expressed broadly*. That is, emotions must affect everything about the character: the way it moves, the way it talks, the expression on its face. An underlying assumption here is that the purpose of a good character is to clearly communicate its thoughts, feelings, and personality to the audience (or, in the case of interactive characters, the user). By expressing emotion in only the character's face, for example, artists find it harder to communicate than if the whole character is used to express the emotion. Thomas and Johnston have this to say on the subject [Thomas 1981]:

If a scene calls for showing tense emotions such as anguish, scorn, bitterness, or envy with only facial expression, the animator will be quite limited. But if the story is built so that the character reveals these feelings in what he does and how he does it... the scenes can be gripping and entertaining. (p. 482)
The expression must be captured throughout the whole body as well as in the face. (p. 443) [Emphasis in the original.]

One producer at Disney's insisted that if a character said he felt a certain way, that was all that was needed.... But it does not work like that. It is not enough simply to proclaim that a character is mad or worried or impatient. There must be business to support the statement and a situation in which he can demonstrate these emotions if the audience is to be convinced that it is so. (p. 387)

Finally, the arts remind us that *the goal is believable emotions, not realistic emotions*. Artists will often want to create characters that are exaggerated or "larger than life," which is at odds with achieving realism. Also, animated characters can be believable, even though they are clearly unrealistic. Some characters may seem quite realistic; others will be wildly unrealistic, but they can all be *believable* in the artistic sense of the word. Even Walt Disney had trouble expressing it. Again, from
[Thomas81]:

There was some confusion among the animators when Walt first asked for more realism then criticized the result because it was not exaggerated enough... When Walt asked for realism, he wanted a caricature of realism. One artist analysed it correctly when he said, "I don't think he meant 'realism.' I think he

meant something that was more convincing, that made a bigger contact with people, and he just said 'realism' because 'real' things do... (p. 66)

Thomas and Johnston, however, are not unclear on the issue [Thomas 1981]:

It should be believable, but not realistic.... Tell your story through the broad cartoon characters rather than the "straight" ones. There is no way to animate strong-enough attitudes, feelings, or expressions on realistic characters to get the communication you should have. The more real, the less latitude for clear communication. (p. 375, emphasis added)

To summarize, the four important lessons to draw from the arts are:

- Emotions are important for creating believable characters.
- Emotions need to be specific to the character in question.
- Emotions need to be expressed broadly.
- Emotions must be believable but may not always be realistic.

3.1.2 Personality

As quoted in [Silva and al, 2001], personality plays an important role as well because it has a strong influence on emotion expression, social behaviour and decision-making, as because it evokes expectations that keep the user involved in the story [Reilly and Bates, 1992] and [Rousseau and Hayes-Roth, 1997].

Personality must be strongly and reliably represented, and it may be expressed in several ways, e.g. text, facial expression, graphics and so on. The importance of personality has been well understood by cartoonists and filmmakers such as Walt Disney © and Warner Bros. ©, who obtained impressive results with some of their characters (e.g., Goofy and Bugs Bunny, respectively). In order to create illusion of life, synthetic actors must also exhibit proper *emotions* according to the facts of the story plot. Expressing emotions, such as ache, joy or rage, increases the possibility of agent's reactions and personality traits being perceived by the users [Reilly, 1996].

Quoted from Thomas and Johnston [Thomas and Johnston, 1981]:
"From the earliest days, it has been the portrayal of emotions that has given the Disney characters the illusion of life."
If the character does not react emotionally to events, if they don't care, then neither will we. [Bates, 1994].

3.2 Few models of personality for synthetic actors

Let us now see a review of some models for representing personality for synthetic actors. What can be a character, a personality and how can we represent it in a "program", in a believable agent?

Here, we are not going to give a formal definition of personality because in fact, this definition can change and depends on the context in which the agent is designed for. Indeed, the way to represent personality can be different in each application. So, the purpose here is to give a general idea of what a synthetic personality looks like. First, in section 3.2.1, we are going to explain the OCC model which can be considered as a standard for emotion synthesis. Then, section 3.2.2 introduces a set of tools, collectively called "Em" that support artists in the creation of believable emotional agents. The section 3.2.3 will talk about the personalities of the Altruist and of the Spiteful. These personalities can be characterized by assigning different priorities to pre-defined "General Goals". In section 3.2.4, the explained model distinguishes *the personality traits, the moods, and the attitudes* that enable the specification of the character at psychological and social levels. Section 3.2.5 identifies few features linked to the design of synthetic actor for long-term computer games. Then, we are going to finish this summary by the section 3.2.6 that displays some functional representations for personality model.

3.2.1 The OCC model

The OCC [Ortony and al, 1988] model, for instance, has established itself as the standard model for emotion synthesis. A large number of studies employed the OCC model to generate emotions for their embodied character. Many developers of such characters believe that the OCC model will be they ever need to equip their character with emotions. In fact, emotions are an essential part of the believability of embodied characters which interact with humans [Elliot, 1992; Koda, 1996; O'Reilly, 1996]. Characters need an emotion model to synthesize emotions and express them. The emotion model should enable the character to argue about emotions the way humans do. An event that upset humans, for example, the loss of money, should also upset the character [Bartneck C., 2002].

Ortony, Clore and Collins have developed a computational emotion model. It specifies 22 categories of emotions based on valenced reactions to situations constructed either as being goal relevant events, as acts of an accountable agent (including itself), or as attractive or unattractive object. It also offers a structure for the variables, such as likelihood of an event or the familiarity of an object, which determines the intensity of the emotion types. It contains a sufficient level of complexity and detail to cover most situations an emotional interface character might have to deal with. The OCC model is complex and so the best way to approach it is by discussing its features in terms of the process that characters follow from the initial categorization of an event to the resulting behaviour of the character. The process can be split in five phases:

1. **Classification:** In the classification phase the character evaluates an event, action or object, resulting in information on what emotional categories are affected.
2. **Quantification:** In the quantification phase, the character calculates the intensities of the affected emotional categories.
3. **Interaction:** The classification and quantification define the emotional value of a certain event, action or object. This emotional value will interact with the current emotional categories of the character.

4. **Mapping:** The OCC model distinguishes 22 emotional categories. These need to be mapped to a possibly lower number of different emotional expressions.
5. **Expression:** The emotional state can be expressed directly through facial expression and can influence the behaviour of the character.

Moreover and in order to finish this brief presentation, OCC uses goals, standards and attitudes. For example, imagine that the user give bananas to the character. So, for it, it needs to have a goal "staying alive" to which the bananas contribute (GOALS). It needs to know what to expect from the user (only knowing that the user does not have to hand out bananas every other minute the character will feel admiration) (STANDARDS). It needs to know that it likes bananas (ATTITUDES). These three notions are for a large part domain dependent.

This assumption implies that consistency is an important factor for the believability of a character [Ortony, 2003]. If bananas make the character happy now then it should continue to do so in the future.

So, this model specifies how events, agents and objects from the universe are appraised according to an individual's goals, standards and attitudes. These three (partly domain-dependent) parameters determine the 'personality' of the individual.

3.2.2 Model for believable emotional agent:

Now let us see how [Reilly, 1996] has constructed his model for believable emotional agents. We are not going to explain every features of his work in details but we are going to try to give some samples and overviews of his general architecture. Indeed, the work achieved by Reilly has already been explained in the previous chapter. Here, and like we are going to explain it, we will focus on his "emotion architecture".

First, he introduces a set of tools, collectively called "Em". In fact, he created a number of tools that support artists in the creation of believable emotional agents: a framework (or architecture) for building emotional agents and a specific system built within this architecture which provides reasonable default emotional processing. The architecture allows many unrealistic (but possibly interesting) agents to be built and the default emotional processing, though informed by the psychology literature, has been tailored to meet a specific artistic end.

The first tool provided for the artists is an *emotion architecture* that sits within a larger agent architecture. The emotion architecture determines the boundaries of what is and is not possible for the agent builder to create in terms of emotional agents. For example, the architecture determines what inputs are available to the agent builder for determining which emotions the agent will have. If the agent builder didn't have access to the agent's goals, it would be impossible to create emotions based on those goals.

The first things to notice in Fig.3.1 are the **Inputs from Tok**. These inputs are used to decide when an agent should react emotionally. The Em architecture provides a wide range of inputs to the artist-defined rules which determine what emotions the agent will display.

The **Emotion Generators** box represents this set of rules (called emotion generators) that take the set of inputs and produce a set of **Emotion Structures**. These rules are written in the Hap language. An example emotion generation rule is the following: when an agent has a goal failure and the goal has importance X, generate an emotion structure of type distress and with intensity X. Emotion structures have a type (e.g., fear), an intensity (e.g., 7 out of 10), possibly a direction (e.g., Bart), and a cause (e.g., Bart is threatening to beat me up).

A set of **Emotion Storage Functions** takes the emotion structures as they are created and puts them into an **Emotion Type Hierarchy**. Emotion structures are placed in this hierarchy based on what kinds of effects they will have, with higher-level nodes representing more general effects and lower-level nodes

representing more specific effects. For example, the hierarchy might have a distress type that represents general forms of distress expression, like frowning, crying, and moving slowly. Below that type might be subtypes, such as grief, homesickness, and lovesickness, which inherit the general effects of their common parent, but that also have more specific means of expression as well, such as thinking about home when homesick. Each type in the hierarchy (e.g., distress) has an intensity associated with it that is a function on the intensities of the emotion structures of that type. The way that the intensities of the emotion structures are combined is determined by a set of **Emotion Combination Functions**.

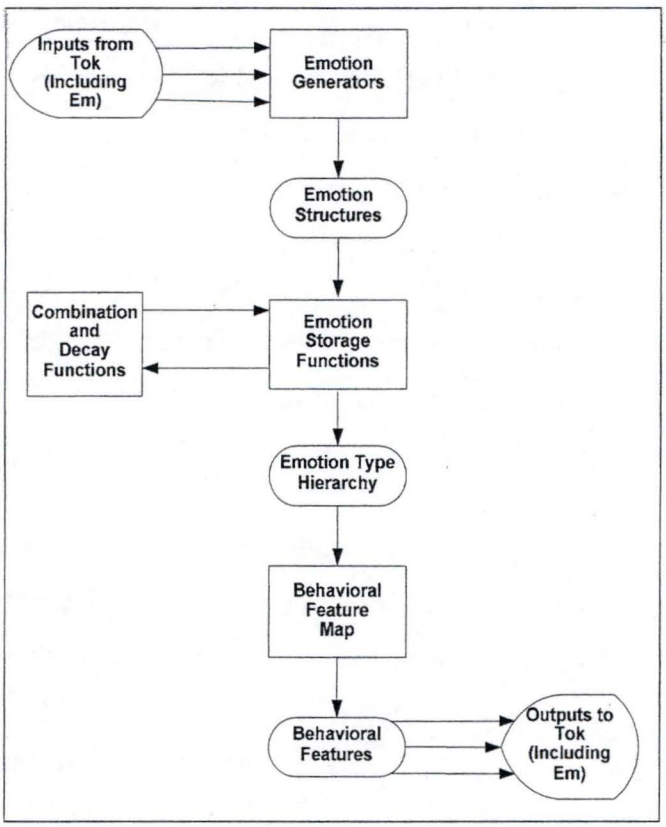


Figure 3-1: The Em Architecture [Reilly, 1996]

The intensity of the emotion structures will decay over time at a rate specified by the artist in the **Emotion Decay Functions**. Each structure can have its own decay function if desired (e.g., anger from being insulted decays slower

than other emotion structures), or decay functions can be defined at the emotion-type level (e.g., all anger emotion structures decay slowly) or over all emotions (i.e., all emotions decay at the same rate).

The emotion structures are mapped into **Behavioural Features** via a **Behavioural Feature Map**. This arbitrary mapping is written in Hap. It is the behavioural features, and not the emotion structures, that directly affect behaviour. The final components of the Em architecture are the **Outputs to Tok**. The behavioural features are able to affect a number of different aspects of the agent's processing.

The second tool: The Em architecture provides the structure an artist will use when creating emotional characters, but none of the content. For artistic reasons, the architecture is quite flexible and so it can be hard to know how to begin. For instance, artists have a large amount of flexibility in determining how to map inputs to emotion structures, but coming up with a good mapping is still a hard problem. Similarly, determining the speed of decay, the proper combination functions, a good type hierarchy, and the other decisions to be made are all difficult tasks.

Because of this, Reilly has not only provided an architecture for creating emotional agents but he has also created a default emotion system. An emotion system is essentially a filling in of the architecture to provide behaviours that are reasonable. It may be useful to think of the emotion architecture as a programming language for writing programs, which control the emotions of characters and the default system, as a sample program. Artists will probably still want to write their own programs, but if they already have one to modify or at least to learn from, creating other programs should be simplified.

Let us see a brief overview of what the default emotion system contains:

• A set of emotion generators based on the cognitive emotion model of Ortony and al. [Ortony 1988] is provided. These generators will create 24 different types of emotion structures.

• These structures are stored in a default emotion type hierarchy.

• The intensities of the emotion structures are combined according to a default combination function and they decay using one of two default decay rates.

• A default behavioural-feature map creates 33 types of behavioural features based on the current emotional state.

• Em provides some default mappings from behavioural features into changes in the way the agent behaves. It is generally difficult to provide default behaviour based on the behavioural features because of the wide variety of characters artists might be working on. However, he has created default mappings from behavioural features to a limited set of emotional effects that he has found are somewhat general, including the body state and changes in attitudes towards other agents. He has also built a number of general functions that artists can use for computing the priorities of goals based on input from the behavioural features.

Reilly gives some advices to the artists to know how to use the Em architecture and the default system. It is not of our concern here. However, since the author uses a terminology, which is specific to his work and which contains implicit assumptions about his approach, it is worthy to explain them briefly.

- **Emotion architecture (Em).** A general framework for creating *emotion systems* for particular agents.

- **Emotion system.** The emotional makeup of a particular agent. For example, getting violently angry when insulted is an aspect of an agent's emotion system. How important an agent feels certain goals are is another aspect. So are the appraisals for determining how likely a given goal is to succeed. Specific emotional episodes (e.g., getting angry) are a result of the emotion system but are considered part of the *emotion state.*

- **Emotion generators.** Anything that produces an *emotion structure* is an emotion generator. A demon that creates joy whenever one of an agent's important goals succeeds is an emotion generator. Such a demon will also be referred to as an *emotion generation rule.*

- **Emotion structures.** Structures representing specific emotional experiences. In the Em architecture, emotion structures include type, intensity, cause, and directional information. For example, if Sam is threatening to beat up Bart, then an emotion generator in Bart might generate an emotion structure of type FEAR. The intensity might be 7 (on a 1-10 scale). The direction represents who the fear is of: Sam. In this case, the cause is the fact that Bart doesn't want to be beaten up and may be represented by a goal structure that represents that information.

- **Emotional state.** The current set of all emotion structures present in an agent. For example, a child being threatened by a bully may feel fear, sadness, and anger (to varying degrees) at the same time.

- **Emotion types**. Emotion types represent sets of emotions that are similar in how they affect the behaviour of the agent. For instance, distress might have a number of subtypes, including grief and homesickness. All distress emotions will share certain effects, like slowing movement and frowning. Other effects are specific to the subtypes, like thinking about home when homesick but not when grieving.

- **Behavioural Features (BFs).** Intermediate structures between emotion structures and *emotion effects*. BFs allow emotion effects to correspond to the current emotional structures without being tied directly to those structures. BFs have the same components as emotion structures: type, intensity, direction, and cause.

- **Behavioural Feature Map.** A mapping (typically) from emotion structures to behavioural features. The BF Map might map emotion structures of type ANGER to behavioural features of type AGGRESSIVE. The mapping can take non-emotional inputs into account, such as creating an aggressive BF to help achieve a goal instead of for emotional reasons.

- **Emotion effects.** Any way that the agent changes based on the current behavioural features. An effect might be frowning or adding a revenge

goal to the motivation system. Effects can also occur in the emotion system itself.

- **Attitudes.** Attitudes are long-term feelings about people or objects. For example, Bill might like Samantha and dislike green beans. These attitudes might give rise to emotion structures but are not themselves emotion structures. Attitudes may change over time but tend to change slowly (as opposed to emotion structures).

- **Moods.** Moods are abstractions based on the current emotional state. For instance, he will refer to a character with a large number of "positive" emotion structures (e.g., joy, hope) as being in a good mood. Moods are determined by the behavioural feature map, so different emotion systems can be built to support different kinds of moods beyond "good" and "bad."

3.2.3 Personality biased behaviour:

Another approach viewed by [2] to realize believable agents that can perform personality biased behaviours in a virtual environment where they can interact with a user through a simple text-based interface is:

The author focused on how to model the agent's personalities by means of an integration of generative and reactive planning techniques which allow to automatically producing personality-biased behaviour.

Some psychological approaches (e.g. [Pervin, 1989]) consider personality as a coherent pattern of kinds of behaviour and interaction with the environment across multiple contexts. Here, typical goals and preferences over actions and plans play a major role in characterizing personalities.

Starting from an AI work [Carbonell, 1980] which has proposed to model human personality by means of goal trees, they assume that personality can be basically defined as (1) a cluster of "General Goals" (G-GOALS) with different

priorities, and (2) a set of goal-based preferences over actions and plans for achieving goals, where preferences concern the relationships between its high priority goals and the side-effects of actions and plans, and can be automatically computed by a planning algorithm biased by suitable heuristics. This model is more deeply described and compared with other approaches in [Rizzo, 1998]. In fact, this work is focused on the comparison of two personalities, the Altruist and the Spiteful.

In order to characterize these personalities, the work on human motivation by [Ford, 1992] is a good source of inspiration. Indeed, he has developed a psychologically-based taxonomy of goals at an abstract and decontextualized level of analysis. From that taxonomy, the following goals have been chosen:

- *Resource provision*: Giving approval, support, assistance, advice, or validation to others. Avoiding selfish or uncaring behaviour.
- *Social responsibility*: Keeping interpersonal commitments, meeting social role obligations, and conforming to social and moral rules. Avoiding social transgressions and unethical or illegal conduct.
- *Belongingness*: Building or maintaining attachments, friendships, intimacy, or a sense of community. Avoiding feelings of social isolation or separateness.
- *Hostility*: Causing troubles to others. Preventing others from getting what they want.

The personalities of the Altruist and of the Spiteful can be characterized by assigning different priorities to the G-GOALS above. The Altruist sincerely cares about others, and is willing to help them, even at its own disadvantage: so, its most important G-GOALS are *Resource Provision* (the Altruist wants to give others what they need) and *Belongingness* (which is useful to characterize the benevolence and kindness that usually accompany an Altruist's behaviour), while a low priority is assigned to *Hostility* (the Altruist, being inclined to help others to get what they need, does not want to cause them any trouble). By contrast, the Spiteful wants to damage others, by interfering with their plans, refusing to help,

or playing tricks on them: so, its most important G-GOAL is *Hostility* (the Spiteful agent wants others to get into trouble), while little importance is assigned to the G-GOAL of *Social Responsibility* (the Spiteful agent does not care very much about social rules) and to the G-GOAL of *Resource Provision* (since it tries to cause troubles to others, the Spiteful agent is unlikely to help them).

With this work we see how it is possible to realize agents that show different personalities of help-giving, by modelling each of the latter as a cluster of goals having different priorities and of preferences over operators and plans.

3.2.4 A Social-Psychological Model:

This model (displayed in [Rousseau and Hayes-Roth, 1997]) provides synthetic agents that behave like intelligent actors portraying fictive characters by improvising their behaviour without detailed planning in a manner like human actors involved in an improvised performance [Hayes-Roth and van Gent 1996]. Human improvisers and the synthetic actors spontaneously and cooperatively generate their stories and portray their characters at performance time following directions from sources such as the audience or the users, a predefined, abstract scenario [Sweet 1978], or other actors [Johnstone 1992].

So, as we are going to see, the authors propose a social-psychological model for fictive characters portrayed by synthetic actors. This model allows the designer of a Virtual Theater (see Fig. 3-3, section3.2.5) application to define a character's personality, moods, and interpersonal relationships in a way that influences the synthetic actor's behaviour consistently without being completely predictable. In fact, the particularity here is that the agent behaviour is improvised and the model covers speech acts and non-verbal actions.

In this model, we can distinguish *the personality traits, the moods, and the attitudes* that enable the specification of the character at psychological and social levels.

Personality traits correspond to patterns of behaviour and modes of thinking that determine a person's adjustment to the environment [Atkinson et al. 1983]. Personality is a persistent characteristic that changes little and slowly, if at all, over time. Personality traits are quite recognizable through an individual's behaviour, because they are what make this person act differently from others. According to [Reilly and Bates, 1995], it is the most important aspect to add in synthetic actors' social behaviours to make them believable.

The personality model is based on two types of theories: *the trait theories* and *the social learning theories*. In trait theories, we assume that traits predispose people to behave consistently, no matter the situation. But in social learning theories, we assume that a personality is modified by each situation viewed as a learning experience through observation and reinforcement. For example, a waiter may be shy with women, but self-confident with men because of his past.

Moods correspond to emotions such as happiness and anger, or sensations from physical needs such as fatigue, hunger and thirst [Rousseau and Hayes-Roth 1996]. Emotions are triggered by events. Sensations are event independent. Both are quite variable over time.

Here the moods are divided into two categories: the *self-oriented moods*, and the *agent-oriented moods*. The self-oriented moods, such as happiness, pride and thirst, are not directed toward other individuals. The agent-oriented moods, such as anger and reproach, are directed toward other individuals, but do not characterize the relationship with those characters. The distinction between self-oriented and agent-oriented moods is important, because an individual may have very different feelings for a particular character from the ones that he or she has in general. For instance, a character may be rather happy in general, yet remain angry at another individual because of what this character did to him or her.

Attitudes characterize an interpersonal relationship [Moulin and Rousseau 1996]. They can vary over time, but can also be very stable, depending on the nature of the relationship. They correspond to the essence of the relationship,

which distinguish them from the agent-oriented moods. Some examples of attitudes include status, degree of sympathy, and trust.

3.2.5 Synthetic Actor Model for Long-Term Computer Games

In long-term games, such as strategy and adventure ones, it is necessary to guarantee both personality stability and reactive emotional responses, which may be contradictory. This part proposes a new Synthetic Actor model that tightly connects emotions and social attitudes to personality, providing a long-term coherent behaviour [Silva and al, 2001]. This will briefly be introduced in the following section.

In order to further increase realism in the design of computer games, intelligent agent's paradigm and techniques are used to model the game's characters, giving them the capability of expressing emotions, personality and goal driven behaviour.

Several synthetic actor models have been already proposed but the problem is that they do not fully meet the requirements of long-term games whose duration usually exceeds a dozen hours. It is the case of, for example, The OZ Project [Bates and al., 1992] (see Fig. 3-2) and The Virtual Theater Project [Rousseau and Hayes-Roth, 1997] (see Fig. 3-3) which have a common characteristic. They both use personality traits that can be modified abruptly according to mood and attitude changes. However, in long-term consideration, characters must exhibit consistent and persistent personality. This is the case of cartoon characters whose caricature-based, never-changing personality helps to improve the credibility level. The psychology community shares this interpretation of personality as a structural long-term characteristic. In fact, some psychologists view personality as a complex human factor that influences the emotion, perception of the environment and cognitive processes [Allport, 1927]. Personality may slightly change, but this process is a long-term one, not directly connected to transitory changes on the emotional state.

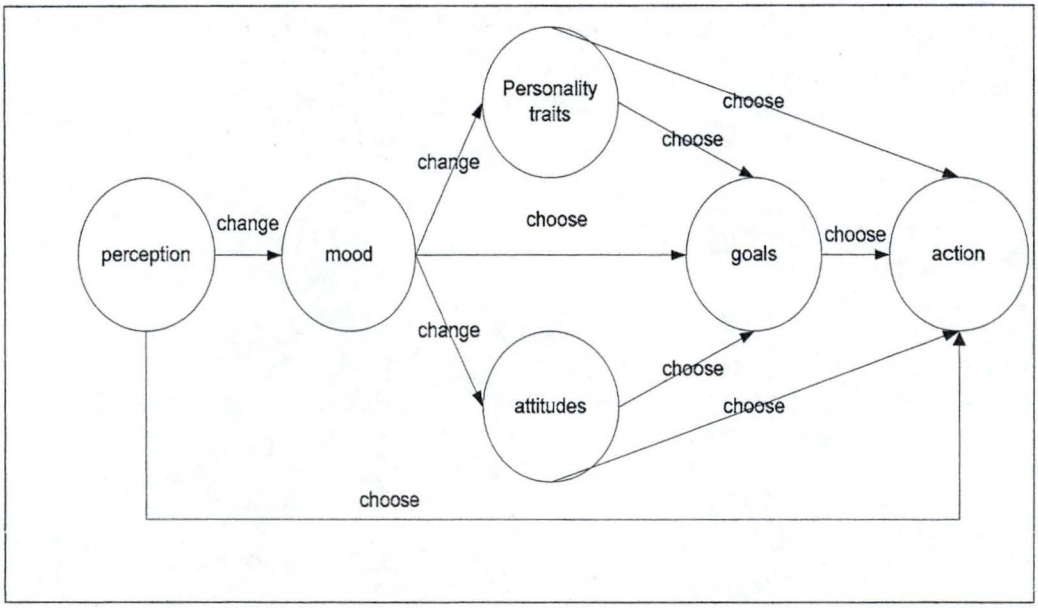


Figure 3-2: The OZ Project model

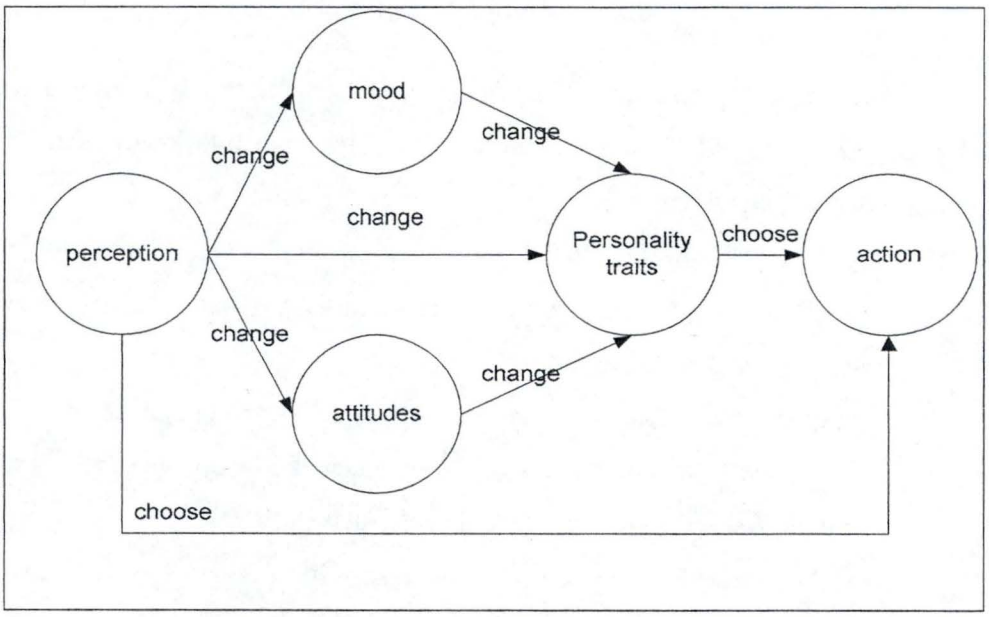


Figure 3-3: Virtual Theater Project model

The GULL [Rizzo and al., 1997] (see Fig. 3-4) project model conceives personality as a time-invariant element whose definition does not depend on other components, such as emotion or attitudes. This model guarantees behaviour

coherence, which improves believability in long-term applications. However, by excluding mood and attitudes, this model neglects two important factors in creating the illusion of life.

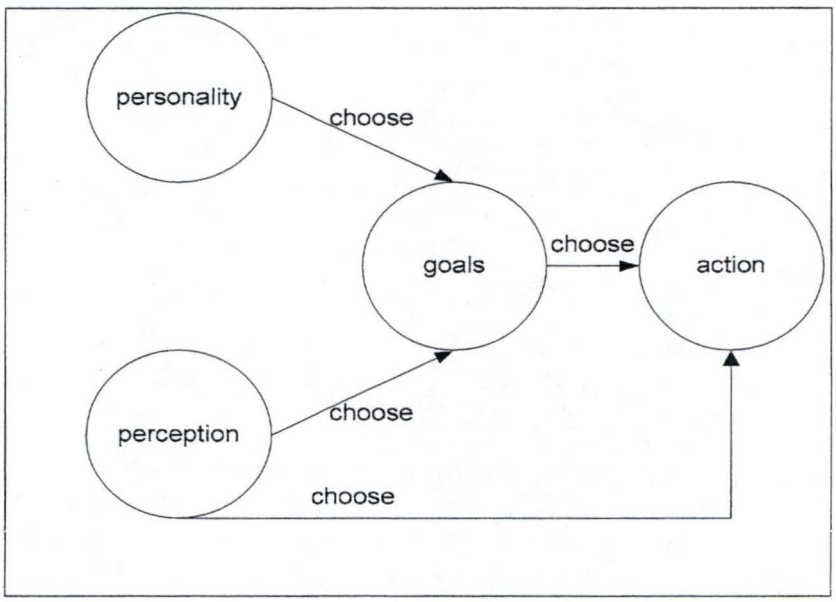


Figure 3-4: GULL Project model

The personality model created here is based on the Big Five Theory [Howard and Howard, 2000], according to which the five main classes of personality traits are:

- **Friendliness** that indicates whether an individual is naturally nice or hostile towards the others.
- **Openness** that concerns agent's interests. High openness refers to characters with lots of shallow interests, whereas low openness refers to an Synthetic Actor (SA) with few and deep interests;
- **Emotional stability** that defines the sensibility level of SAs with respect to their perception. Resilient SAs are only bothered by strong stimuli, whereas reactive SAs are bothered by a greater variety of stimuli;
- **Agreeableness** that concern characters self-confidence. High agreeableness applies to SAs that follow social conventions, whereas low

agreeableness describes an SA that, in the extreme, only follows its own rules;

- **Consciousness** refers to the number of goals a character is focused on. Highly conscientious SAs focus on few goals and exhibit the self-discipline associated with such focus. Low conscientiousness refers to an SA that pursues a larger number of goals, exhibiting spontaneity.

As we can see in Fig. 3-5, personality influences mood and attitude values by defining their initial and default values. For instance, let us suppose that the user assigns a friendly personality to a given actor. This assignment automatically sets a high value for the happiness feature of this actor. The recurrent arrows in mood and attitudes of Fig. 3-5 indicate a time dependent regulation mechanism which forces their values to get closer to the default ones. For instance, if a tragic event induces temporary sadness on the friendly actor, then, as time goes by, its happiness will go back to the high level. Furthermore, this actor will not be angry with another actor for a long time, no matter what the latter has done.

Personality also influences mood and attitudes by amplifying or attenuating the impact of the environment perceptual data. For instance, if a coward synthetic soldier meets a strong enemy, its fear level increases more than if it was a courageous one. In this model, the first reasoning step is responsible for updating the SA mood and attitudes according to sensory perception and personality trait values. The second step concerns the selection of the goals to be achieved as a function of mood, attitudes and personality. Finally, taking into account the perceptual data and the goals, the actions or plans are selected.

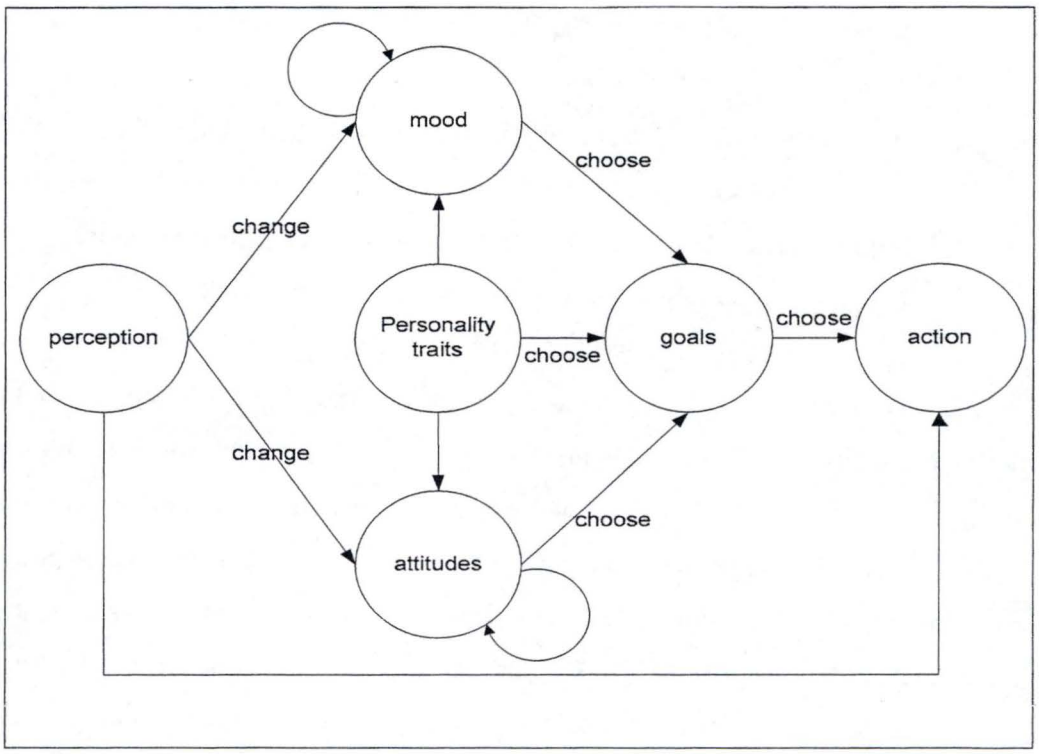


Figure 3-5: Personality model for Long-Term Computer Games

These steps are carried out by production rules stored in three knowledge bases [Russell and Norvig, 1995]. In the first knowledge base, rule preconditions take into account perceptual information and personality traits, whereas rule conclusions determine how mood and attitudes vary. An example of such rules is: "*if* the agent is sensitive (personality trait) and it witnesses its friend killing another agent (sensing), *then* it will become very sad (mood), and the friendship (attitude) toward its friend will be diminished".

The second knowledge base contains rules, whose preconditions concern mood, attitudes, and personality traits to determine the goal. Some examples of these rules are: "*if* the agent is fatigued (physical mood) or bored (emotion) but it is perseverant (personality trait), *then* it will keep its current goals" and "*if* the agent does not trust the other agents, but it is optimistic (personality trait), *then* it will be ready to negotiate with them".

The rules of the third knowledge base simply instantiate the goal into concrete actions according to the perceptual data. For instance, "*If* an agent whose goal is

to negotiate with agents of a given ethnic group meets another agent of this group, *then* it will negotiate with the latter".

This work has proposed an original model for synthetic actors, whose goal is to provide both long-term personality cohesion and short-term emotional reactions.

3.2.6 Conversational personality model:

As presented in [Egges and al, 2003] where the authors introduce a generic model for describing and updating the parameters related to emotional behaviour for conversational virtual humans, we can consider personality as follow.

An individual is an entity that is constantly changing. So, when the authors speak of an individual, they always refer to it relative to a time t. An individual has a personality and an emotional state (without taking yet mood into consideration). The model based on this assumption is called PE. The personality is constant and initialized with a set of values on t = 0. The emotional state is dynamic and it is initialized to 0 at t = 0. Thus they define I_t (the abstract entity that represents the individual at a time t) as a tuple (p, e_t), where p represents the personality and e_t represents the emotional state at time t.

There exist many personality models, each of them consisting of a set of dimensions, where every dimension is a specific property of the personality. Generalizing from all the existing theories, they assume that a personality has n dimensions, where each dimension is represented by a value in the interval [0, 1]. A value of 0 corresponds to an absence of the dimension in the personality; a value of 1 corresponds to a maximum presence of the dimension in the personality. The personality p of an individual can then be represented by the following vector:

$$p^T = [a_1 \dots a_n] \cdot \forall i \in [1, n] : a_i \in [0, 1] \qquad (1)$$

Emotional state has a similar structure as personality, but it changes over time. The emotional state is a set of emotions that have a certain intensity. They define the emotional state e_t as an m-dimensional vector, where all m emotion intensities are represented by a value in the interval [0, 1]. A value of 0 corresponds to an absence of the emotion; a value of 1 corresponds to a maximum intensity of the emotion. This vector is given as follows:

$$e_t^T = \begin{cases} \left[\beta_1 \ldots \beta_m\right], \forall i \in [1, m] : \beta_i \in [0, 1] & \text{if } t > 0 \\ 0 & \text{if } t = 0 \end{cases} \quad (2)$$

Furthermore, they define an emotional state history w_t that contains all emotional
states until e_t, thus:

$$\omega_t = \langle e_0, e_1, \ldots, e_t \rangle \quad (3)$$

The PME model is an extended version including the moods of the PE model,. They now define the individual I_t as a triple (p, m_t, e_t), where m_t represents the mood at a time t. They define mood as a rather static state of being, that is less static than personality and less fluent than emotions [Kshirsagar and Magnenat-Thalmann, 2002]. Mood can be one-dimensional (being in a good or a bad mood) or perhaps multi-dimensional (feeling in love, being paranoid). However, to increase generality, they will provide for a possibility of having multiple mood dimensions. We define a mood dimension as a value in the interval [-1, 1]. Supposing that there are k mood dimensions, the mood can be described as follows:

$$m_t^T = \begin{cases} \left[\gamma_1 \ldots \gamma_k\right], \forall i \in [1, k] : \gamma_i \in [-1, 1] & \text{if } t > 0 \\ 0 & \text{if } t = 0 \end{cases} \quad (4)$$

Just like for the emotional state, there is also a history of mood τ_t that contains the moods m_0 until m_t:,

$$\sigma_t = (m_0, m_1, \ldots, m_t) \tag{5}$$

From perceptive input, an appraisal model such as OCC will obtain emotional information. This information is then used to update the mood and emotional state. The emotional information is defined as a *desired change in emotion intensity* for each emotion, defined by a value in the interval [0, 1]. The emotion information vector a (or *emotion influence*) contains the desired change of intensity for each of the m emotions:

$$a^T = [\delta_1 \ldots \delta_m], \forall i \in [1..m] : \delta_i \in [0..1] \tag{6}$$

The emotional state can then be updated using a function $\Psi_e(p, \omega_t, a)$. This function calculates, based on the personality p, the current emotional state history w_t and the emotion influence a, the change of the emotional state. A second part of the emotion update is done by another function: $\Omega_e(p, \omega_t)$ which represents internal change (such as a decay of the emotional state). Given these two components, the new emotional state e_{t+1} can be calculated as follows:

$$e_{t+1} = e_t + \Psi_e(p, \omega_t, a) + \Omega_e(p, \omega_t) \tag{7}$$

In the PME model (which includes the mood), the update process slightly changes. When an emotion influence has to be processed, the update now happens in two steps. The first step consists in updating the mood; the second step consists of updating the emotional state. The mood is updated by a function $\Psi_m(p, \omega_t, \sigma_t, a)$ that calculates the mood change, based on the personality, the emotional state history, the mood history and the emotion influence. The mood is internally updated using a function $\Omega_m(p, \omega_t, \sigma_t)$. Thus the new mood m_{t+1} can be calculated as follows:

$$m_{t+1} = m_t + \Psi_m(p, \omega_t, \sigma_t, a) + \Omega_m(p, \omega_t, \sigma_t) \tag{8}$$

The emotional state can then be updated by an extended function Ψ'_e that also takes into account the mood history and the new mood. The internal emotion update, which now also takes mood into account, is defined as $\Omega'_e(p, \omega_t, \sigma_{t+1})$. The new emotion update function is given by the next formula:

$$e_{t+1} = e_t + \Psi'_e(p, \omega_t, \sigma_{t+1}, a) + \Omega'_e(p, \omega_t, \sigma_{t+1}) \qquad (9)$$

3.3 Conclusion:

The existing models for synthetic actors are, in majority, based on the classic agent architecture. In this architecture, a sensorial system continuously acquires environment information; an inference mechanism selects actions considering the sensorial data, and the agent's goals and internal state; and an ensemble of effectors executes actions [Russell and Norvig, 1995]. The *internal state*, which can be a representation of both the external environment and the agent's mood, is updated at each inference cycle. The main components explicitly represented in almost all synthetic actor models are the following:

- ***Goals*** that denote desirable states of the environment (e.g., all cities of the world were conquered; all enigmas were solved, etc.).
- ***Mood*** that designates the emotional and physical state of the SA. In most of the current systems, mood affects the goals and/or action selection. For instance, the feeling of fear (c'est un mood la peur??) can make a synthetic actor give up its main goal and run away.
- ***Attitudes*** that are interpersonal relations between synthetic actors or a synthetic actor and the environment. For instance, a synthetic actor representing a cat will act differently depending on what it has seen: a dog or a mouse.
- ***Personality*** that is modelled as personal behaviour patterns. For instance, a lazy agent will escape its obligations. In the proposed

> synthetic actor architectures, personality is usually modelled as an array of traits to which numerical values are assigned. For example, friendly/unfriendly.

Despite these shared features, there are several differences among the current synthetic actor models. Most of these differences concern how personality influences (or is influenced by) other variables (mood, attitudes and goals); how all these variables are influenced by the environment events; and how all these variables influence the decision-making (action selection) process [Silva and al, 2001].

Chapter 4: General presentation of our chatterbot

4.0 Introduction

An experimental E-market virtual world technology provides an immersive, complete trading environment. Traders are represented as avatars that interact with each other, and have access to market data and general information which are delivered by data and text mining machinery. To enrich this essentially social market place, synthetic bots have been constructed. They too are represented by avatars and are indistinguishable from avatars that represent human traders. Although they provide little more than "idle chatter" they fulfil a key role by enriching the social fabric. They are built on deep models of character, and are clearly individuals. They acquire their information with text and data mining machinery which continually scans market data and general financial news feeds.

In this chapter, we are going to present our work achieved in the E-market Research Group at UTS.

What we have done is a program, a robot or more exactly a chatterbot. The first goal identified for him, is to provide casual, but context-related conversation with anybody who chooses to chat with him. In order to realise that in a good way, our chatterbot has been given a character, a personality which allows him to be seen like a real actor, like a human. All of these points will be explain in this chapter. First, in section 4.1, let us detail the context in which the bot plays. Indeed, the context in which we can understand the work done is dual. We can imagine our bot in a virtual E-market place and, inside this virtual place, in a Stock Exchange chat room context. Most of the time in this chapter, the virtual E-

market context will be used and displayed because it is this one that embodied the main goals of our work. In section 4.2, we are going to identify the main issues relating to the design of synthetic character. Then, section 4.3 will talk about data mining approach which helps to extract some knowledge for the bot. Section 4.4 will introduce the theoretical features of our chatterbot. In section 4.5, we will consider the fact that our bot can be seen as an agent in certain circumstances. Section 4.6 will talk about Ethical and moral consideration of creating such believable chatterbot and in order to finish this section, we will identify few possible uses of our program.

4.1 Context

4.1.1 The virtual E-market context

There has been an enormous amount of work in the development of electronic commerce systems since the late 1990's. Despite the "dot com" crash this work continues, both in business-to-consumer commerce and in business-to-business commerce. Markets play a central role in economies, both real and virtual. One interesting feature of the development of electronic markets has been the depersonalization of market environments.

The majority of on-line trading today is conducted by "filling in pro formas" on computer screens, and by "clicking buttons". This has created a trading atmosphere that is far removed from the vital atmosphere of traditional trading floors. What used to be the trading floor of the Sydney Stock Exchange is now a restaurant, although the trading prices are still displayed on the original board. All of this sits uncomfortably with work in microeconomics that has demonstrated both theoretically and in practice, that a vital trading environment provides a positive influence on liquidity and so on clearing prices too. See, for example, the work of Paul Milgrom, from his seminal "linkage principle" to his recent work [Milgrom, 2004].

This issue is being addressed by the eMarkets Research Group at the University of Technology, Sydney. There, virtual world's technology, based on Adobe Atmosphere is being used to construct virtual trading environments that may be "hooked onto" real exchanges. That work aims to produce trading environments that are immersive and complete. They are "immersive" in that: an avatar that may move freely through those virtual trading areas in which it is certified represents each real player. They are complete in that each real player has ready access to all the information that he requires, including general information extracted from news feeds.

These "future generation trading environments" include avatars that represent virtual traders too. Work led by Professor John Debenham is building trading agents that operate in these information-rich environments [Debenham, 2003]. Members of the UTS group are constructing an electronic institution framework that "sits beneath" the virtual environment. The design of this framework is influenced by the Islander framework developed in Barcelona [Esteva and al, 2002]. It contains a rich variety of actor classes to manage the warrant made by trading agents. Work directed by the leader of the eMarkets Group, Dr Simeon Simoff, is applying unstructured data mining techniques to tap real-time information flows so as to deliver timely information, at the required granularity [Simoff, 2002].

If these trading environments are to provide the rich social experience that is found in a real, live exchange then they should also contain other classes of actors in addition to essential classes described above. For example, anonymous agents that can provide intelligent casual conversation—such as one might have with a stranger in an elevator. To have a positive role in an eMarket environment, such anonymous agents should be informed of recent news, particularly the news related to the market where the agent is situated (e.g. financial news, news from major stock exchanges, news about related companies). If so then they provide a dual social function. First, they are agents to whom one may "say anything" (social actor), and, second, they are a source of information (informer) that is additional, and perhaps orthogonal, to that provided directly to each trading agent as it has been specified.

Fig 4-1 shows a screenshot of a market scenario in a virtual world. The avatar that is "standing" behind the table represents this agent. The avatar with its "back to the camera" is the avatar representing the agent on the workstation from which the "photo" was taken.

Figure 4-1: A screen shot of one of the eMarkets, including the "chatterbot" avatars.[2]

4.1.2 The Stock Exchange chat room context:

This context is presented here only to understand the way we have built our program. It will allow us to see why we have used such technology and why we have chosen such design. In fact, in this part, we don't really have to consider our bot represented by an avatar in a virtual E-market environment as the aim pursued here is to construct a conversational agent.

The context is the one where a synthetic actor (our chatterbot) is designed to be like any human. It has to speak, to talk in a way like humans do. It is capable of giving any financial information you asked for. But the real purpose of such chatterbot is to detect people giving wrong, and so illegal, advice on Stock Exchange's chat room. Then, when people are detected, the program launches an

[2] A demonstration that contains one of these synthetic characters is available at: http://research.it.uts.edu.au/emarkets/ — first click on "virtual worlds" under "themes and technologies" and then click on "here". To run the demonstration you will need to install the Adobe Atmosphere plugin.

independent agent (agent can be launched as much as there are "bad advisers" detected) in order to talk with this "bad adviser" person and get information contact from him.

That is all, and this is one of the contexts in which our bot has been built. Of course, this program doesn't use the real Stock Exchange chat room, and that is why we have implemented a fictive chat room interface which allows some testing. Moreover, this context allows us to introduce an ethical and moral consideration of creating program endowed by personality. This part will be explained a bit later (section 4.6).

4.2 Few issues:

The following issues have to be addressed in the design of any synthetic character:

- its appearance
- its mannerisms
- its sphere of knowledge
- its interaction modes
- its interaction sequences

and how all of these fit together.

Appearance is a key issue in initiating contact but is not addressed here. But, a 3D application has been developed (see section 4.9). Mannerisms are equally important—a researcher associated with the UTS group is investigating the impact of facial expressions on interaction. This is a major issue that is beyond the scope of this discussion. At present the avatar in Fig. 4-1 "shuffles around".

An agent's sphere of knowledge is crucial to the value that may be derived by interacting with it. We have developed extensive machinery, based on unstructured data mining techniques, that provides our agent with a dynamic

information base of current and breaking news across a wide range of financial issues. For example, background news is extracted from on-line editions of the Australian Financial Review. This will be described in detail in the data mining part. The agent uses this news to report facts and *not* to give advice. If it did so then it could be liable to legal action by the Australian Securities and Investment Commission ASIC!

Our agent's interaction modes are presently restricted to passive, one-to-one interaction. They are passive in that these agents do not initiate interaction. We have yet to deal effectively with the problem that occurs when a third party attempts to "barge in" on an interaction.

The interaction sequences are triggered by another agent's utterance. The machinery that manages this is designed to work convincingly for interactions of up to around ten exchanges only. Our agents are designed to be "strangers" and no more—their role is simply to "toss in potentially valuable gossip". The management of the interaction sequences is described in section 4.4.3.

4.3 Data mining approach:

This part has been achieved by Benjamin Dosquet and Xavier Magnant during their internship at UTS.

Text mining as opposed to classical data mining techniques is used here to extract knowledge from *unstructured* text streams.

The agents do not know in advance what will be "said" to them. So they need to be able to access a wide range of information concerning the world's and the market's situation, much of which may not be used. This large amount of information is extracted from different sources on the web, including news pages, financial data and all the exchanges that are part of the virtual environment. Within a particular interaction, the agent's response is driven only by keywords

used by his chattermate — just as a human may have done. Despite this simple response mechanism, the agent's dialogue should not appear to be generated by a program. This has partly been achieved by using a "semantic network" of keywords extracted from the input streams. Words are given a relational weight calculated against others. This can be seen in fig 4-2. By analysing which words are related to those used by the chattermate, the agent expresses *himself* and, to some extent, is able to follow the thread of the discussion. All of this takes place in real time.

In addition to the semantic network, the agent has a personal character and background history (described in section 4.4.3), which influences his appearance and the way in which he employs the keywords in his responses. The agent's knowledge is divided into two classes: general and specific. Specific information is contained in a particular data structure (product info table in fig 4.2). In further work, both classes of information will be extracted on demand if they have not previously been pre-fetched.

To extract the general knowledge, a suite of extractors are activated that fetch predefined text streams from the web. After an extractor has retrieved its raw data (for example, all headlines from sportingnews.com in order to be able to speak about current results in a particular sport), the raw data is manipulated by an intermediate module that converts it into structured data ready for classical data mining techniques. Fig 4.3 shows the general structure of the involved components.

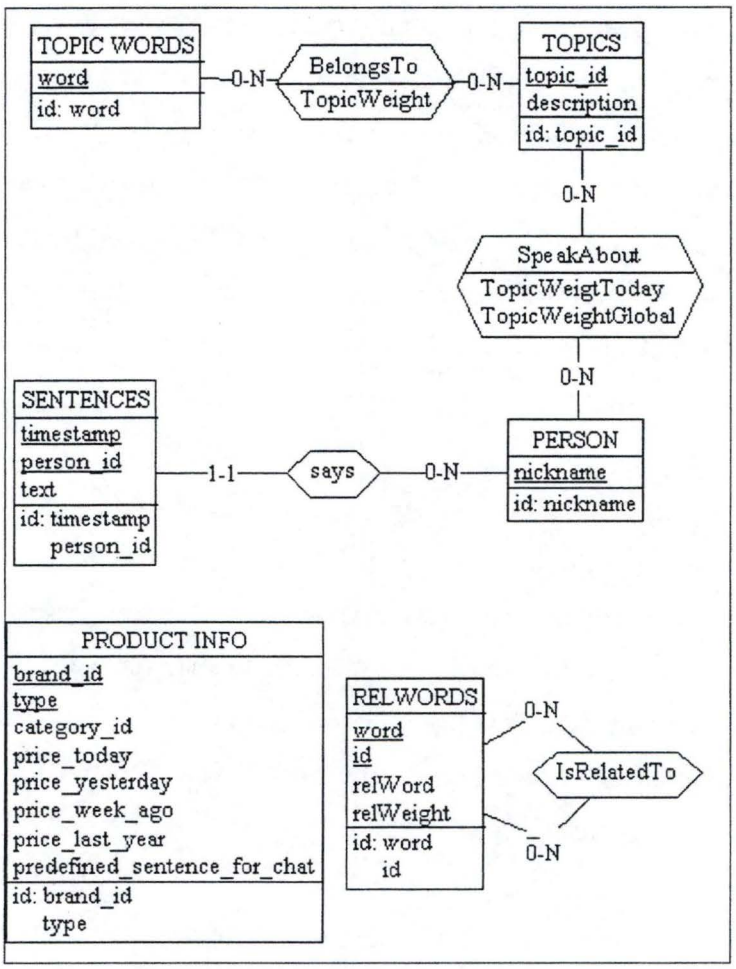


Figure 4-2: Simplified Entity-Association Diagram of the Database

An example shows how all of this fits together. Consider an agent that is situated near the virtual Sydney Stock Exchange. Its pre-fetched, general knowledge may contain: the previous days closing prices for all ASX [Australian Stock Exchange] listed stock, the current (every 5 minutes) values for the major international indices (including: Dow, Nasdaq, DAX, Hang Seng, FTSE etc), headline news only from the Australian Financial Review (which reports some general news as well as financial news). This raw data is fed into the semantic net. This net is the agent's "initial knowledge base". For example, if an interaction uses a news headline then, in further work, the full news story will be retrieved, indexed and added to the net. This will be normally achieved quite quickly and the augmented net will be available in time to construct the subsequent response.

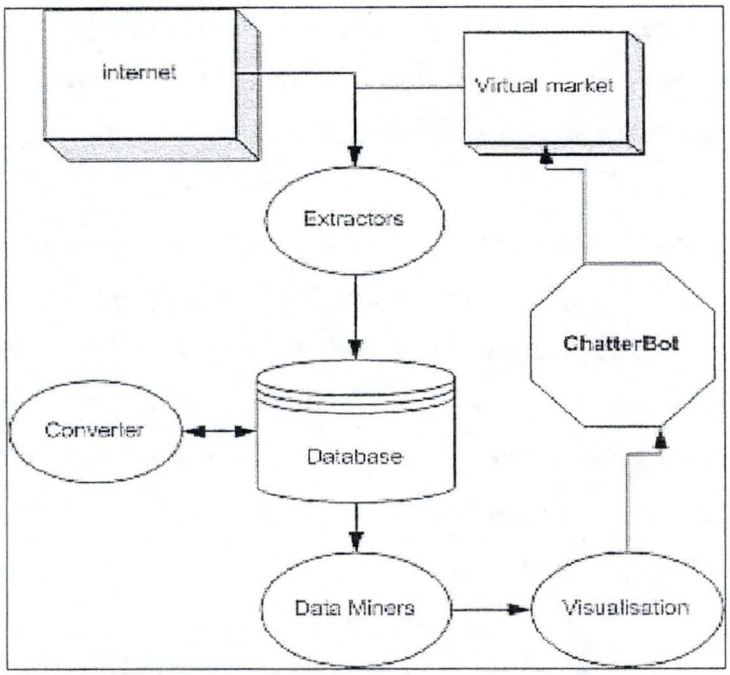


Figure 4-3: Structure of the Data Mining module

As another example, suppose an interaction concerns the closing price of a stock of a company that does not appear in the headlines represented in the initial net. This, in a close future work, will trigger a search for information about that company using a set of standard web sources and general search engines. All of this takes time. If the subsequent interaction appears to stay with this theme then the agent responds along the lines: "Hey, give me a minute — I'll see what I can find."

A third module then applies mining techniques to discover what is then called *information*. The desired information is retrieved from this information base on demand by the chatterbot. Most of the information required by each agent is related to a specific topic. Related information is identified and temporarily stored in the central database. In order to be usable by an agent, the information undergoes a visualisation process. Then the agent is in a position to use these retrieved phrases to construct its response.

Specific knowledge is pre-fetched from one of a set of pre-specified websites. For information about chocolate for example, he fetches the price history, the available brands and different packaging. This raw data is fed into the

semantic net which is then used to guide the interaction. If the interaction contains the word "Neuhaus", then because the semantic net is now augmented with data about chocolate, Neuhaus is recognized as being a chocolate brand.

The previous simple example demonstrates how the agent identifies the theme of an interaction. This simple device works well as long as the theme does not alter. If the theme changes unexpectedly quite amusing responses may occur. For example, suppose that Neuhaus is also the name of an Australian race horse. Consider an interaction in which the chattermate asks about the wholesale price of chocolate and then asks about the betting odds on Neuhaus in the 3:00 race at Randwick (a Sydney race course).

So the agent may occasionally respond with crazy. This is not seen to be a weakness. As interaction sequences tend to be brief and single-themed, this phenomenon is rare. But when it does occur it is not necessarily a bad thing[3].

4.4 Theoretical features of our chatterbot

In this section, we will talk about the personality dimension, then about the consistency problem concerning this personality dimension. The high level design of our chatterbot will be displayed in detail in section 4.4.3. Section 4.4.4 will explain some aspects of the technical design and we will finish this section by a high level representation of our personality model.

4.4.1 The personality dimension:

What is different from other already existing chatterbot? The principal feature of our chatterbot is the contribution of a "specific personality model" for synthetic actors. In fact, each agent is clearly individual in a quite long period of time in order to leave the impression of talking to another actor at each

[3] See, for more details, the thesis achieved by Benjamin Dosquet and Xavier Magnant.

interaction. This particularity is provided here by the agent's character. The first decision that is made when an agent is created is to select its character using a semi-random process that ensures that multiple instances of the agent in close virtual proximity have identifiably different characters. This one can be seen and considered as a set of three components -personality traits, the moods (self oriented) and the attitudes (with others).

Personality traits correspond to patterns of behaviour and modes of thinking that determine a person's adjustment to the environment [Atkinson and al. 1983].

Moods correspond to emotions such as happiness and anger, or sensations from physical needs such as fatigue, hunger and thirst [Rousseau and Hayes-Roth 1996].

Attitudes characterize an interpersonal relationship [Moulin and Rousseau, 1996].

However, our agent's character is essentially composed of the personality traits and the moods, but, attitudes which correspond to the essence of the relationship are left behind and hence, are not considered here. In fact, the character's dimensions (here, the personality traits) that we have selected are intended specifically for a finance-based environment. They are:

- Politeness
- Dynamism
- Optimism
- Self-confidence

The meaning of each of these dimensions is (moderately) independent of the others:

- Politeness means the use of polite words, phrases and forms
- Dynamism is the tendency to react rapidly, succinctly and vigorously
- Optimism here means a tendency to use up-beat phrases and the tendency not to use negations
- Self-confidence here means the tendency to respond with declarative statements rather than tentative propositions or questions.

The selection of the character of an agent determines: its appearance (i.e.: a male or a female) and the style of its dialogue. Future plans will address the avatars mannerisms. The selection of agent's character provides an underlying unifying framework for how the agent appears and behaves, and ensures that multiple instances of the agent appear to be different.

The selection on an agent's character does not alone determine its behaviour. Each agent's behaviour is further determined by its moods that vary slowly but constantly. The dimensions of moods that we have identified are:

- Happiness
- Sympathy

Now, let us consider a few examples of dialogues modulated by the character dimension. The following examples were generated with various character and mood parameter settings. In the first interaction the agent's character is very dynamic and self-confident, and is in a rather bad mood:

Guest: what's in the news today?
Agent: *An earthquake in Japan.*
Guest: yes, but in the business news?
Agent: *The Nasdaq is 1792 and falling.*
Guest: I would have preferred the Australian business news…
Agent: *In Australian news the chairman of BHP has resigned.*

In the second example, the agent's character is shy and retiring (he has a very low value of self-confidence). He is also very polite and is in a good mood:

Guest: what's in the news today?
Agent: *Uh.., what sort of news do you want?*
Guest: the business news…
Agent: *Excuse-me, but I have a large amount of business news. Do you want international or national?*

Guest: the international news!!!
Agent: *Today the Nasdaq is 1792 and falling ... Is there anything particular that you want to know?*

In the third example the agent is optimistic, self-confident and is in a good mood:

Guest: what's in the news today?
Agent: *There was an earthquake in Japan.*
Guest: could you give me the business news too?
Agent: *Of course. The Nasdaq is falling.*
Guest: damn! I have a lot of shares in it.
Agent: *You have a lot of shares?*

In the fourth example, the agent's character is non-dynamic, impolite and he is in a bad mood:

Guest: what's in the news today?
Agent: *What news?!*
Guest: business.
Agent: *The Nasdaq is 1792 and falling.*
Guest: and in Australia?
Agent: *How much more do you want to know? The chairman of BHP has resigned.*

These four small scenarios show how, by basing the interaction sequences, first on a fixed character and second on a variable mood, we can generate interactions that are not predictable, but still informative.

4.4.2 For the sake of consistency

The main problem identified here is staying consistent whatever happens! Indeed, since the character is composed of six dimensions (including the moods), we have to fit all these dimensions together in order to give the possibility of having believable agents.

Let us consider two simple examples, the first one shows a consistent possible character for an agent (in Fig. 4.4). On the other hand, the second is clearly a bad and inconsistent character for an agent (in Fig. 4.5).

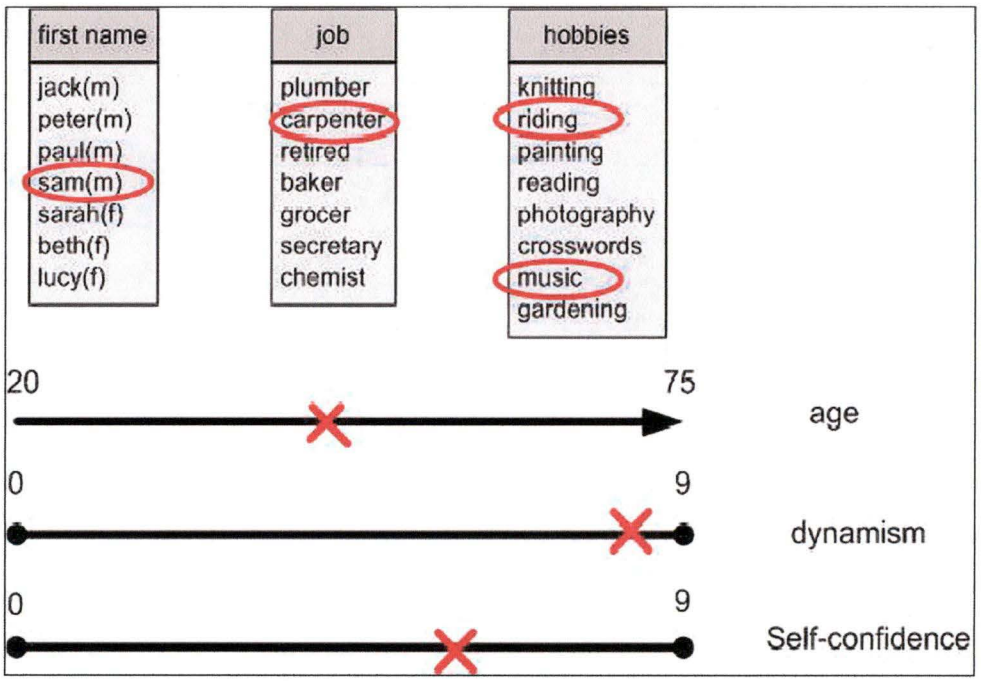


Figure 4-4: Possible consistent example for an agent's character

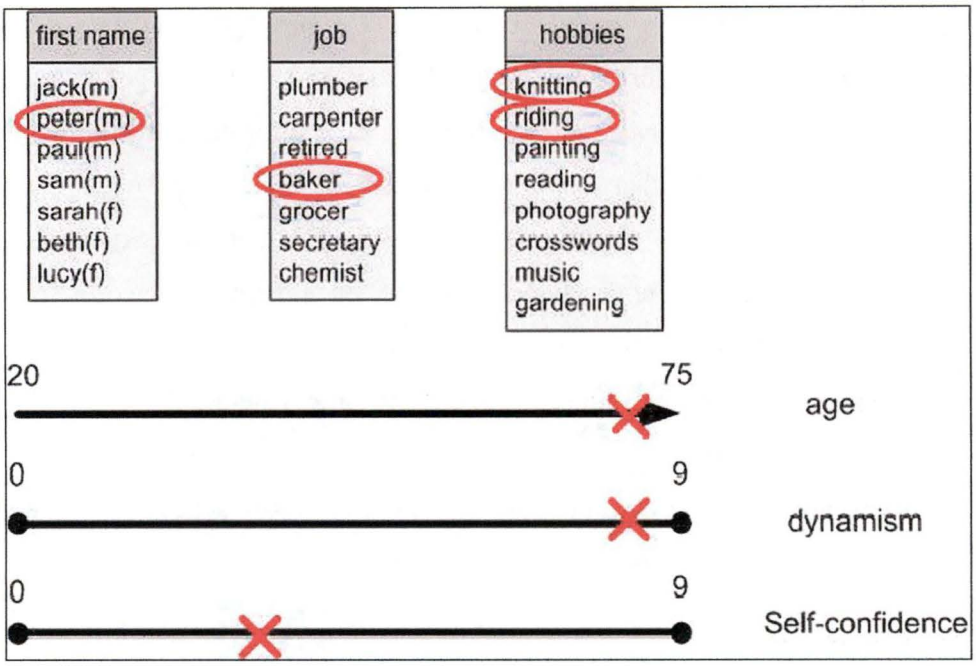


Figure 4-5: Possible inconsistent example for an agent's character

But, it is not all, for this consistency issue, other problems must be dealt with : any agent has to have, at any time, a different description and also he has to act differently from other agents. Here, the random process takes part of differentiation. Moreover, we can consider that: if one agent's reactions are modulated by his own personality and if every agent has his unique personality, then every agent will have a different behaviour. In fact, when the personality of the agent is randomly created in the initialization part, there is something like a "guarantor" of consistency which checks for the character six possible values.

4.4.3 Explanation of the design

First, let us briefly introduce our chatterbot's architecture in order to understand how an answer can be produced by the bot and how the personality dimension influences these answers.

The way in which the whole chatterbot apparatus fits together is shown in Fig. 4.6. There it is represented by an avatar in the virtual market place. The chattermate will also be represented by an avatar in the market place and will

initiate the interaction. The initial sequence is captured by the *interaction manager* which extracts the principal keywords from it—what is judged to be the crucial topic is called the *genus*. It has a data base of informal phrases such as "What's new?" which, in that example, is linked to the genus "news". This resource has been initially constructed by hand, but will be extended by interpreting real data. The *ontology module* relates the extracted keywords to the semantic net that resides in the indexed *information base*. For example, if the initial sequence contains the phrase "news ?? Japan" [where "??" can be any word], the ontology module makes the link between "an earthquake in Tokyo" (stored in the semantic net) and the word "Japan". Now the agent has something to say. The response is formed in the *marked-up phrase base*.

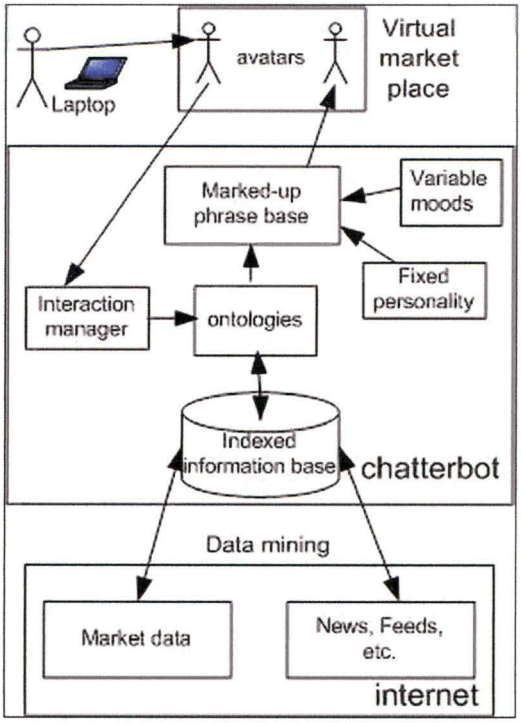


Figure 4-6: General structure of the chatterbot

In the example above the agent was able to identify what the chattermate was asking for. This will not always be possible. This is the situation that humans find themselves in when visiting a foreign country where they have a limited grasp of the local language. The agent has two basic mind sets. First it assumes that the chattermate is asking for some information and it attempts to identify and

provide that information. This assumption may be incorrect, and even if it is correct the agent may not be able to understand it. In the second mind set the agent acts like a traditional ELIZA-bot and fishes for more information.

Now, let us consider what role is played by the character dimension. The "self-confidence" character dimension is implemented as a reluctance to change to the two mind-sets. A self-confident agent will attempt to guess a response using more information coming from the web; an agent with low self-confidence will be inclined to ask for more information that it knows (information about itself for example).

The marked-up phrase base is a table of response templates that are "marked up" with indices linked to the values of the agent's character and mood parameters. The value of each dimension of character is represented by an integer in the range [0, 9] with a neutral value of 4. Likewise for the mood's dimension. So, for each of the six dimensions, there are 10 possible values. That gives 10^6 possible different states of an agent! This is serious over-kill, for the purposes described here—it has been included to support future work. At present the agent's mood changes in time dependently of the interaction—this varies the interaction and helps to prevent it from appearing "automatic". The character of the agent remains constant.

The mark-ups in the phrase base are constants that represent archetypal responses for character only. The varying mood is expressed by occasionally including additional words. So the character mark-up is a two digit integer to base 10 which represent respectively the politeness and the optimism. The role of the "self-confidence" dimension is described above. With regard to the dynamism, this dimension is managed (like the other dimensions) by a random process in the agent creation's part and its role is identified by the tendency of reacting rapidly, succinctly and in a way of speaking as well. A dynamic agent will reply quicker than a less dynamic one. The phrase base is in two separate sections: one for each mind-set.

A sample from the phrase base for the first mind-set is shown in Fig. 4.7. This phrase base is used when the agent already has an answer (perhaps a news headline) to the chattermate's request. This answer will be a declarative statement denoted by <INFO1>. The phrase bases have been constructed by hand which is rather tedious. The agent's character does not restrict it to its mark-up. The responses are selected at random but are weighted towards the agent's type.

In Fig. 4.7, two possible outputs are shown for the four types of mark-up. These two figures mean that the sentence, which will be selected for the agent's reply, will correspond to an agent with a quite big value of politeness and a quite small value for optimism. In fact, the random figure, obtained after the agent' character creation, for these two dimensions will be put in correspondence with the four possible mark-up values (in fact, there are more than four possible mark-up values but those represented here allows a minimal cover) in order to find the good sentence corresponding to the character type of the agent. Example: let us consider that the random figure for the politeness and the optimism is 63, then this value corresponds (with a correspondence method) to a type of 72.

Genus	Mark-up	Answer	Outputs
news	22	<INFO1>	What the hell, #INFO1# This is a headline from {www.}{abc.com}: #INFO1#
	27		I was just checking the news and it appears that #INFO1#
	72		I heard that #INFO1#. The newspapers say that #INFO1#
	77		On {the} {www.}abc.com {web}site you [could|can] read about #INFO1#

Figure 4-7: Sample phrase base for first mind-set.

The second mind-set is more complicated. Given the eMarket context, if an agent is unable to respond with an answer from its semantic net then it delivers general financial news and all about its identity. For example, it does not pretend to "be an Eliza" and generate sentences such as "I am sorry to hear you are ?X?."! Although the marked-up phrase base employs the Eliza approach. Separate from the genus, the interaction manager identifies a pattern in the input phrase. Here

the patterns are constructed around financial and identity interaction. Fig. 4.8 contains simple examples for the four types of character.

Word	Mark-up	Pattern	Outputs
age	22		I'm #AGE#
	27		I've been on Earth for #AGE# long years
	72		{Well, }actually I'm #AGE#[years old|.] Yeah, I'm #AGE#{ years old}
shares	77		What company? What sort of shares?

Figure 4-8: Sample phrase base second mind-set.[4]

To sum up, the interaction manager extracts the genus, key phrases and a pattern from the input. The genus may be unknown. The ontologies are used to connect the key phrases with the contents of the semantic net. All being well, this identifies an answer. The answer, if there is one, is fed into the marked-up phrase base to generate the response. The style of the response is governed by the agent's fixed character and variable mood. This framework is not unlike the way that a human answers a question. Once the answer is composed, it is "said" by the chatterbot's avatar.

4.4.4 Technical design

Technologies behind our chatterbot

We have developed our parts of the program independently of the rest of it. So the part that manages all the treatment of the conversation, the creation of the sentences, the personality of the agent, etc. could be reused in other programs. According to this aim of reusability, an interface between the Agency package and the rest of the program (whatever the program is) has been developed. This interface is the InOut package. Using the operations defined in this package, the

[4] The pattern column is present for further work. It will be used to sharpen the answers replied by the bot

other parts of the program can "work" with the Agency package and so the creation of an agent and the creation of answers by an agent.

Xml

We decided to use XML technology for several reasons. That is what we are going to explain in this part. But before, let us consider briefly what XML is.

What is XML?

The XML standard (XML 1.0) has been defined by the W3C (World Wide Web Consortium) institution in February 1998. XML has been developed in order to publish document on the web. Since it tends to be used more and more for exchange format between applications and more generally, between information systems [Walsh, 1998].

XML is a markup language for documents containing structured information. Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays (for example, content in a section heading has a different meaning from content in a footnote, which means something different than content in a figure caption or content in a database table, etc.). Almost all documents have some structure.

A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to documents. XML specifies neither semantics nor a tag set. In fact XML is really a meta-language for describing markup languages. In other words, XML provides a facility to define tags and the structural relationships between them. Since there is no predefined tag set, there can't be any preconceived semantics. All of the semantics of an XML document will either be defined by the applications that process them or by style sheets (Extensible Style Language (XSL)).

Which structures are used and why? :

Although there aren't explicit document type declarations (DTD) in ours XML files, this structure definition is, in fact, implicitly present. Indeed, each of ours files is parsed by our application which follows a specific structure in order to correctly extract information.

Actually, we have four different XML files having their own structure. Each file's feature has a specific role, and that is what we are going to explain here. First, we are going to display the implicit DTD that exist behind each files and then explain briefly why such a structure is used.

1. **"ListAnswerOnt.xml"**

DTD :

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ListAnswerOnt (focus+)>
<!ELEMENT focus (genus+)>
<!ATTLIST focus
   thema (news | share | weather) #REQUIRED
>
<!ELEMENT genus (markupPhrases+)>
<!ATTLIST genus
   name CDATA #REQUIRED
>
<!ELEMENT markupPhrases (sentence+)>
<!ATTLIST markupPhrases
   markup (00-99) #REQUIRED
>
<!ELEMENT sentence (#PCDATA)>
```

This structure is used in order to allow an easy and effective extraction of the file. In fact, this file is used to supply a sentence to be answer by the bot according to its personality. This end point is helped by the "markupPhrase" node and its attribute "markup" that determines the character of the bot. (Example: the figure 32 represents a character of a less polite and optimistic bot whereas a 67 figure represents the opposite.)

2. **"ListAnsWithoutOnt.xml"**

DTD :

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ListAnsWithoutOnt (focus+)>
<!ELEMENT focus (markupPhrases+)>
<!ATTLIST focus
   thema CDATA #REQUIRED
>
<!ELEMENT markupPhrases (output+)>
<!ATTLIST markupPhrases
   markup (00-99) #REQUIRED
>
<!ELEMENT output (#PCDATA)>
<!ATTLIST output
   pat CDATA #REQUIRED
>
```

Here, the same explanation as before can be given.

3. **"listWords.xml"**

DTD :

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT listWords (wordPrior+)>
<!ELEMENT priority (#PCDATA)>
<!ELEMENT thema (#PCDATA)>
<!ELEMENT wordPrior (priority, thema)>
<!ATTLIST wordPrior
   word CDATA #REQUIRED
>
```

Here, the "word" attribute of the "wordPrior" node permit to define a specific context. This context is characterised by a specific "thema" (corresponding to the "thema" attribute identified above) and a "priority" that determines in which priority our bot has to take the keyword in consideration.

4. **"job.xml"**

DTD :

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT city (cityName, state)>
<!ELEMENT cityName (#PCDATA)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT hobbie (title)>
<!ATTLIST hobbie
   critSex (f | m | x) #REQUIRED
   critAge (a | m | x | y | z) #REQUIRED
>
<!ELEMENT job (jobTitle)>
<!ATTLIST job
   critSex (f | m | x) #REQUIRED
   critAge (m | x | y) #REQUIRED
>
<!ELEMENT jobTitle (#PCDATA)>
<!ELEMENT liste (job+, name+, hobbie+, city+)>
<!ELEMENT name (first, sex)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT title (#PCDATA)>
```

Here, the "critSex" and "critAge" attribute can have different values that allow a finer selection of a job or a hobby in order to keep consistency. Example: actress is a job that can be made by a woman being whatever age. (See the code documentation in the appendices to have more detail of the value attribute's meaning)

Why XML files?

We decided to use XML technology because it was quite new for us. But it is not the only reason of our choice. The most important reasons are the possibility of our files' evolution, the reuse of our application in a completely different environment. Indeed, our XML module is independent of the rest of the application and the only thing we have to keep is the XML file's structure. In this case, it is quite easy to change the content's file in order to use our bot in another environment (other than a financial environment). Of course, the only files we

should modify are those which are used in the answer creation part. The agent background life's file can in another side be change as well but its impact is not considered (it s not worthy) here. In fact, the agent character's creation is done once in the beginning of the application while the answer's creation is executed each time the bot have to answer.

Moreover, as seen above, stored XML data are self-described by their "implicit" DTD. That is why it is so easy to change content in order to change environment but without changing application. Of course, the opposite of this flexibility is the necessity of decoding XML source with a parser. Indeed, unlike a stable structure that is directly exploitable after reading, the information included in the XML files has to be extracted and decoded by this parser. This is to be explained in the next part.

Extraction methods:

In fact, Java doesn't offer the possibility of processing XML. However, some editors have developed Java API. Currently the most popular APIs available for manipulating the XML Documents are DOM (Document Object Model) and SAX (Simple API for XML).
When comparing DOM and SAX there is one major difference: DOM is a *tree*-based API, whereas SAX is an *event*-based API.

So DOM compiles an XML document into an internal tree structure. Then it allows an application to navigate that tree. SAX, on the other hand, reports parsing events (such as the start and end of elements) directly to the application through callbacks, and does not build an internal tree. Each approach has its limitations: The DOM is memory-consumptive (the bigger the XML data to represent, the bigger the need for resources), the SAX makes it quite complex to access different elements of an XML document besides a linear walk-through.

That is why we chose DOM for our application. Indeed, all ours XML files are not very huge (so it is easy to compile them into an internal tree

structure). Moreover, files concerning agent's background lifer are only parsed and used for the agent's creation. Hence, the memory consumed is not really significant. The following schema shows how an internal tree structure of our "ListAnswerOnt" XML file compile with DOM can be represented.

```
<?xml version="1.0" ?>
<ListAnswerOnt>
    <focus thema="weather">
        <genus name="sunny">
            <markupPhrases markup="22">
                <sentence>The sun will shine #INFO1#. I'll have to take my hat</sentence>
            </markupPhrases>
            <markupPhrases markup="27">
                <sentence>What a pity, it'll be sunny #INFO1# and I've a lot of work to
            do</sentence>
            </markupPhrases>
            <markupPhrases markup="72">
                <sentence>Yeah, there 'll be a lot of sun #INFO1#.</sentence>
            </markupPhrases>
            <markupPhrases markup="77">
                <sentence>I'll probably go wandering #INFO1#, it'll be sunny.</sentence>
            </markupPhrases>
        </genus>
        <genus name="clouds">
            <markupPhrases markup="22">
                <sentence>Clouds, clouds, always clouds... I'm fed up with this weather!
                </sentence>
            </markupPhrases>
            <markupPhrases markup="27">
                <sentence>I really don't like this weather where we cannot say if there 'll be
                rain or not </sentence>
             </markupPhrases>
            <markupPhrases markup="72">
                <sentence>The sky will be cloudy #INFO1#, I hope there 'll be no rain
                </sentence>
            </markupPhrases>
            <markupPhrases markup="77">
                <sentence>The sky 'll be cloudy #INFO1#, we'll probably have some shower
                </sentence>
            </markupPhrases>
        </genus>
    </focus>
</ListAnswerOnt>
```

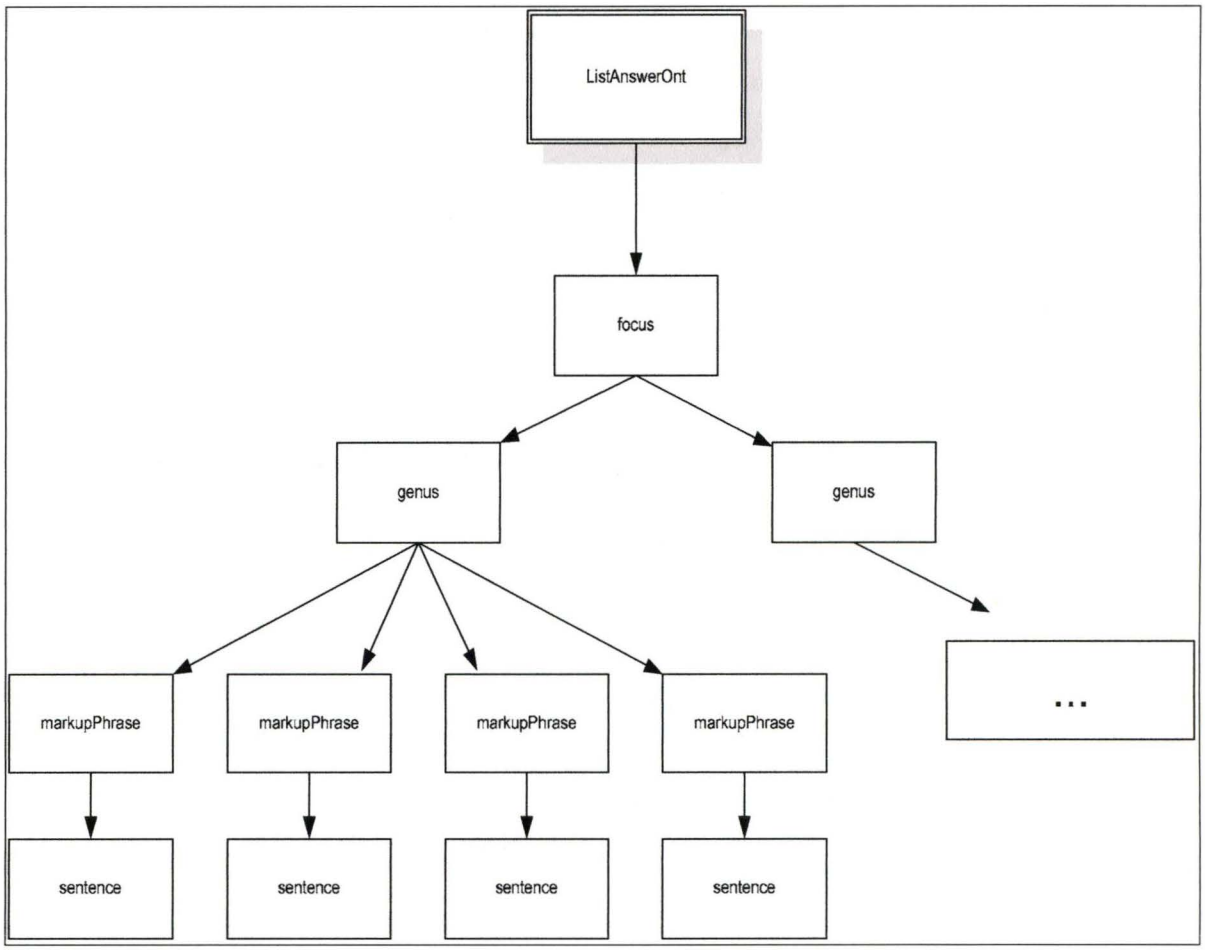


Figure 4-9: DOM tree representation

But it is not all, after having parsed and put each files in an internal tree structure, we don't use DOM method to extract the desire data. In fact, we use XPath. XPath is a language which allows querying, to run through the elements of the document.

XPath use a path notation like it is used in file system and URL to specify and put in correspondence the document's elements. For instance, XPath: `/x/y/z` allows to seek in a document, a root node `x` under which reside a node `y`, under which reside a node `z`.

This instruction return all nodes corresponding to the path structure specified. It is also possible to achieve others queries like:

The instruction / x / y /* returns all nodes behind what ever node y with the parent x. This one, / x / y /[@name = 'a'] correspond to all nodes y which have a parent x and an attribute name with the value a.

4.4.5 How could these XML files generate a conversation

In fact, there are two processes that are both used to generate an answer: the With Ontology process and the Without Ontology process. The last one can be seen as follow.

Each time a keyword is detected in a sentence, the program will take the thema corresponding to this keyword and create an answer in this thema.
Example: *I'd like to know where **you come from**.*

You come from >>> thema: city

We use priority to determine an order of preference between the keywords. So, if there are two keywords in the same sentence, it is the priority that will determine which keyword is the most important.

Once a thema of discussion has been selected, a sentence about this thema must be also selected. This selection is made according to the personality of the agent (especially its optimism and its politeness). One of the sentences that correspond to the personality of the agent will be randomly chosen.

During the accomplishment of the Without Ontology process, the other one is also executed. This means that when the two processes terminate, the program has two possible answers. According to the self-confidence dimension, the bot will choose one of them. A self-confident agent will rather choose the sentence coming from the With Ontology process and a less self-confident one will prefer to use the sentence coming from the Without Ontology process.

After a sentence has been selected, the program will process this sentence to make it personal to the agent. There are five processes that are used to remodel the sentence according to its personality and identity:

- ## is used to put in the sentence some personal details about the identity of the agent (Example: #JOB# >>> carpenter)
- [] is used to put in the sentence one of the different propositions that are between the [] and split by '|'. The choice of the proposition that will be chosen is made according to the dynamism of the agent. The more dynamic is the agent, the more on the right will be the chosen proposition. This process contains also a small random part.
  (Example: [contact|phone|call] >>> *call* for a dynamic agent)
- {} is used for the optional words. What is between the {} will be kept if the agent is happy (process including a random part).
  (Example: Could you {please }help me? >>> *could you help me?* for an unhappy agent)
- abbreviations will be used for the agent that are dynamic
- keyboard errors will also be sometimes used

4.4.6 What can be the model for our personality?

For instance, in a virtual e-market context, we had to build and provide synthetic agents that behave like intelligent actors portraying fictive characters by providing a conversational dialogue without being viewed as something else than a human-being.

Human chatters and our synthetic actor spontaneously and cooperatively generate their dialogues and portray their characters. But, if we want this conversation to work well, the human actor must believe in the bot's behaviour and must be surprised occasionally. Believability and surprise depend on the roles played and on the level of abstraction. For instance, an actor behaving with inconsistency is not very believable. Moreover, if we consider our bot at the second level of abstraction (see section 4.5), the human actor can be more easily surprised in a sense of novelty and adaptability from the bot. What makes a performance believable are its emotional and social aspects [Reilly and Bates 1995].

As built in our "bad bot project" and like described in [Rousseau and Hayes-Roth 1997], we can propose a social-psychological model for fictive characters portrayed by synthetic actors. This model allows us to define character's personality, moods in a way that influences the synthetic actor's behaviour (the way it talks) consistently without being completely predictable. One of the main differences between our project and [Rousseau and Hayes-Roth 1997] is:

- We don't consider the relationships with other bots; indeed, the attitudes are in a way embodied in the personality trait dimension. (There is no conversation with other bots.)
- Our model is essentially a conversational one.
- Moods have still not an effect on personality.
- Values for moods and personality traits are assigned once in the random process. However, in spite of the limited time of the conversations, moods can even change slowly according to the content of these conversations.

So, in our virtual e-market context, our personality model can be seen as follow:

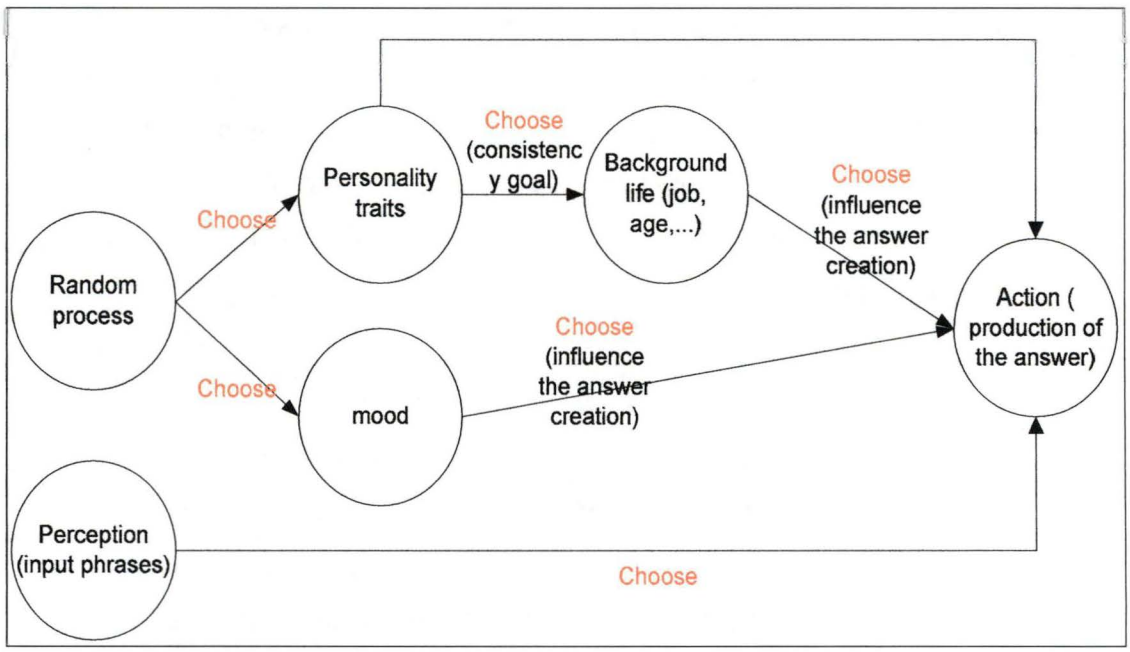


Figure 4-10: Our personality model

Fig. 4.10 depicts the relationship among the main concepts of our model, where personality and mood play a central role. We represent personality as an grouping of traits whose values are defined by a random process.

In the beginning, personality allows choosing a background life for the bot in order to keep consistency for the character portrayed. Then, during the conversation, personality can directly affect the answer's production. For instance, politeness influences the way the bot will reply. Then, mood, background life and perception allow the bot to choose an appropriate sentence to answer.

4.5 Can we consider our chatterbot like an agent?

For considering if the chatterbot described before is an agent, we have introduced (section 1.1.1) the fact that the definition of an agent is very broad. As seen in section 1.1.2, the problem of deciding if a program is an agent or not depends of the level of abstraction in which you observe the program. The program will be inspected regarding three different levels of abstraction. The first one will be the point of view of the user who talks once with the chatterbot; the second one will be the point of view of an user talking with the chatterbot several times; the last one will be at the level of the code. According to the definition of an agent we decided to use, every agent has to be reactive, autonomous, pro-active and persistent

- At the first level of abstraction, the chatterbot cannot be seen as an agent. It is reacting to the sentences written by the user (reactivity), it seems to change its behaviour itself (autonomy), it has also initiated the conversation (pro-activity) but it doesn't seem to last, it disappears from the chat just after a few sentences (no persistency).

- At the second level of abstraction, the chatterbot can be seen as an agent. It always has the reactivity, the autonomy and the pro-activity but also a sort of persistency. Indeed the user can discern that he is speaking each time to the same program although he hasn't chat two times with the same 'agent' (in the sense of character). The user can see, at this level of abstraction that the program is permanent on the chat and so be considered as persistent.
- At the level of the code of the program, we can consider our chatterbot as an agent. It has all the features of an agent according the definition given in the chapter 1. However, some of the features are not very developed; they exist but have only a small impact. The chatterbot obviously contains a good interactivity with the user (as each sentence of the agent is an answer) but the autonomy is limited to start the conversation and decide to close it. At this level of abstraction, we can say that our program is persistent as it permanently scans the conversations. The pro-activeness feature is also implemented as the program permanently observes all that is done on the chat.

As seen above, using our definition, our chatterbot can be considered as an agent. However, using other definitions, our chatterbot sometimes couldn't be seen like an agent. Indeed, some definitions say the adaptability (and so the learning) is an essential feature for having an agent. Today, our chatterbot hasn't the capacity to learn from the experience and so couldn't be considered as an agent by such definitions.

4.5.1 Implementing the learning ability in our chatterbot

As many definitions require the adaptability feature from an agent, we will propose some ways to implement the learning in our program. It is really very hard to make a bot able to learn from a conversation. This is possible only if it understands the meaning of the sentences the user writes. This is, at this point almost impossible. So there must be another way of learning for the chatterbot. The operation that could be made is retaining some information about the

characteristics of the personality that have worked the best. So, if each sentence of a concluding dialog receives some special extra points and if the process of decision of the sentence to be said by the chatterbot includes these points, the chatterbot would at least optimize its behaviour.

Another possibility of improving the program could be by making profiles of the persons the chatterbot speak to and then, using a learning process, react with the character that has the more chances to catch the user. To make profiles of the users, the program could use text mining tools, looking what the persons are speaking about (the program knows their whole conversation in the chat room) and so detect some topics or words that could help to categorize these persons.

4.6 Ethical and moral consideration:

Definition of the concepts:

- Moral impact: An impact always refers to someone or something. In this sense, it is a moral impact if it has a moral prejudice or benefit to any user.
- Bad adviser: a bad adviser is someone, detected by the bot that is giving some information that is judged, according to a bogus detection process, to be advices.

  (The advices identified here are those that are seen as pushing or influencing someone to buy or send something particular on the stock exchange place.)

When designing context-related conversational "agents", it is important to consider the social context in which they will operate. Supposing the Stock Exchange chat room context. According to [Isbister and Hayes-Roth, 1998], good design of intelligent interface agents must take the social role of the agent into account. Quoted in the well respected text in social psychology, "In the service of predictability, people tend to use role schema first, then person schema such as traits to specify a particular version of the role" [p.177, Fiske and Taylor, 1991]. In other words, knowing a person's social role helps us to remember what the

person is like, and also guides our behaviour toward that person. This natural human tendency is going to guide us in our moral approach for our chatterbot. Indeed, what we are going to do now is trying to identify the social role played by our bot. Then we will exanimate whether this or these roles have any "moral impacts".

4.6.1 What are the roles played by our bot?

In a stock exchange context, our bot is designed to fulfil the social role of a social actor like everybody. Indeed, it has a name, an age, a job. It lives in a town, it has hobbies. For example and generated by a random process, we can have a bot whose name is Bart, he is 22 years old and he is a student. He also has a town named Malmedy where he lives. He likes playing badminton, spending time with his friends and of course with his girlfriend. This background life allows us to identify a general social role played by every human-being. But there is more: our bot is also designed to be a new comer to the stock exchange chat room's universe, someone who asks for advices and tries to get some help from the other people connected on the chat. Behind those roles that are easily identifiable, there are two other roles. Our bot is in fact a rules keeper and an information provider as well. Indeed, since the purpose, the "goal" pursued by the bot is to talk to someone, who delivers some "bad information", in order to get information contact from this actor, by this way it acts like something dissuasive. Secondly, it plays also a role of informer. These characters are freely available for interaction and play an important role in delivering current news and information that may not be generally known. They are always well informed about recent events.

So, the roles played by our bot are:

- Social actor
- Beginner in the stock exchange environment
- Rules keeper

- Informer

We are going to analyse the question: Does these roles have any "moral impact"? in the following section.

4.6.2 What could be the moral impact to the society?

A component in our approach and based on [Floridi and Sanders, 2001] is the 'level of abstraction' (LoA) at which an agent is considered to act. The LoA is determined by the way one chooses to describe, analyse and discuss a system and its context. LoA is formalised in the concept of 'interface', which consists of a set of features, the observables. What we are going to explain is that morality may be captured as a 'threshold' defined on the observables in the interface determining the LoA under consideration. An agent is morally good if all its actions respect that threshold; and it is morally bad if some action violates it.

First, let us determine what could be the specific LoA in the right context. The considered context here is a stock exchange chat room where people can log themselves on the chat and begin to talk with other people (synthetic or not). In fact people never know if they are talking to a synthetic actor, they are just aware of this possibility. So the LoA is like as follow: in this LoA, people, users are informed of every applicable rule in the stock exchange chat room. It means that they have to their knowledge the fact that:

- (1) It is not allowed to give some "bad advices" to someone else.
- (2) It is possible that someone (synthetic or not) check the conversation in an aim of dissuasion.

Now, in order to see if the actions executed by the bot are under a certain tolerance's threshold (identified by human agents exercising ethical judgements), we have to look at the possible "moral impact" of the identified roles played by our bot in the LoA considered.

Morality of an action performed by our bot at a given LoA can now be defined in terms of a threshold function. A threshold function at a LoA is a function which, given values for all the observables in the LoA, returns another value. Our bot at that LoA is deemed to be morally good if, for some pre-agreed value (called the tolerance), it maintains a relationship between the observables so that the value of the threshold function at any time does not exceed the tolerance.

As seen above, the roles are:

Being a *social actor* and being like a *beginner* is something common and if we add all the observables coming from those roles to the LoA, our bot can be regarded as acting a role providing a morally good impact if its output maintains the involved user's well-being within an agreed tolerance of his desired well-being. In fact, the involved user is the actor who is talking to the bot, and he will be in a well-being state if, for example, the bot talks to him in a polite manner.

The *rules keeper* (or the dissuader) is a quite important role played by our bot because it allows him to check and listen to the conversations of every user in order to get information contact from a presumed "bad adviser". If we add all the observables coming from this role to the LoA, our bot can be regarded as acting a role providing a morally good impact if its output maintain a certain level of information (in order to avoid any trick) within an agreed tolerance of his desired level. As you can see, the information we are talking about are in fact included in the specified LoA. But we will discuss about that a bit later.

The *informer* consists on giving some information as answer to a question asked by a user. If we add all the observables coming from this role to the LoA, our bot can be regarded as acting a role providing a morally good impact if its output maintain a certain level of accuracy within an agreed tolerance of his accuracy duty.

In fact, they have all a moral impact but what we can see in this case, is that the rules keeper role has the higher one because if we change the LoA to another one which doesn't take in consideration the two information rules identified above ((1),(2)), then the action of rules keeper will provide a morally

bad impact to the society. Indeed, since the LoA won't contain (1) and (2) and including the observables from the role played, it will be clear that the output will be above the threshold of tolerance.

4.7 Possible uses of our program:

E-Market: The chatterbot can be used in the E-Market world developed in Sydney by the E-Market Research Group at UTS (Sydney). This world is a virtual place where people can meet each other to bargain. In this world, an avatar represents each user. The world is divided in different areas of trade. So in one specific area, an user can meet other users wanting to buy or sell the product concerned by this area (wine for example). All these operations require people chatting with each other. So, the idea is that no one knows who is behind an avatar enabling us to introduce our chatterbot in this world.

First, it could be used as a gossip bot, so it could help everyone needing accurate information. Secondly, adding some modules to the bot, it could enable to do a bargain.

We have developed a special application for some of the modules of the chatterbot. Using Adobe Atmosphere, a program which allows building 3-D worlds, and a character Editor, we have designed our chatterbot to be part of a 3-D environment. The character Editor allows the user to create his own character, giving him some movements and some attitudes. We have first generated a character, and then we have given this character a lot of possible movements and attitudes. Doing this, we have generated a character that could have many different ways of acting and so he could potentially play the roles of very different characters. We have added to this multi-character our chatterbot. This step has been done using JavaScript and some small adaptations of the code of the chatterbot. Indeed, in the original version, the chatterbot uses an Access database, but for the E-market version, it uses a MySql database. The interface with the program has also been modified because the format of the answers is quite different. Finally, the program has been modified for accepting the transmission

of some gestures and attitudes to be done by the avatar, i.e. a non-dynamic agent could have a lazy approach and if it said "ciao", the avatar could make a movement with its hand.

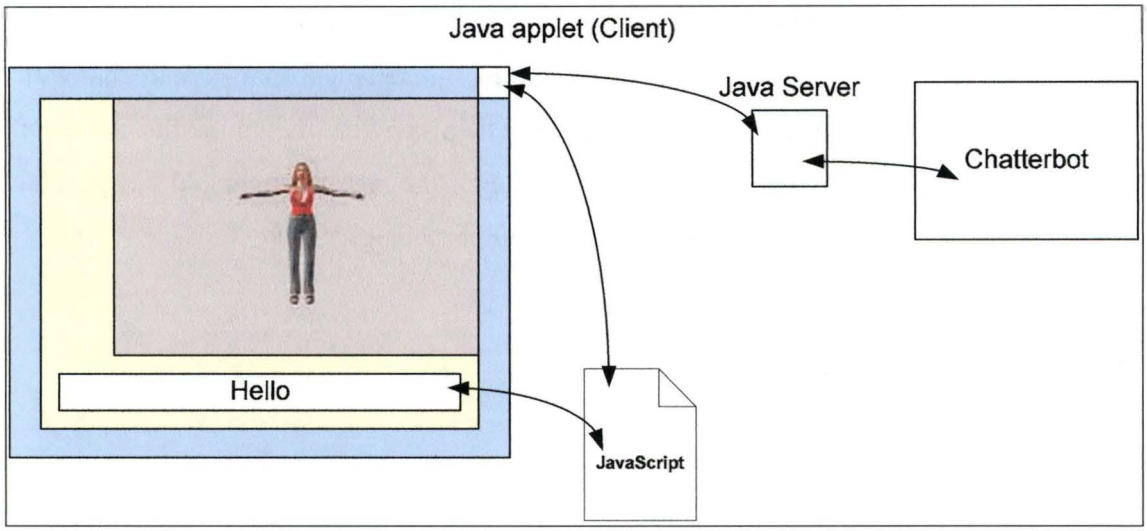


Figure 4-11: architecture of the E-market with the chatterbot

Paedophile: As everyone knows, a huge problem in the chats on Internet is the presence of paedophile looking for young boys or girls. By modifying our program a little bit, it could be able to partly solve this problem. One part of the program could scan the chats, then another part could analyze what is said in these rooms. Afterwards, once a suspicions chatter is detected, a new agent could be sent to the chat room to trap the suspect. For the prospect of attracting paedophiles, the identities of the agents could be identities of children with naïve personalities. Once they have found some details about the guy, they could send that information to the police department or another association.

The program could be used without big modifications in every situation that needs to lure a human-being during a short discussion (if this person has doubts about the nature of the chatterbot, it will be easy for him to know that he doesn't chat with a human-being). The chatterbot can of course be used in every situation that requests a chatterbot (in this case, it could be considered as a gossip bot)

Research on personality: in some areas of research on the influences of the personality in the language, the chatterbot could be a good way of testing if the assumptions made by the searchers are correct or not. The program has been developed in such a way that adapting new mechanisms to express the personality is really easy. So the researchers could implement their mechanisms in the program followed by some tests on a panel of "guinea pigs" that have to chat with different agents generated by the chatterbot then say what they think the character of the agent is. By making a lot of these tests, the researchers could have a real idea of what influences the personality of human-beings.

Conclusion

In the first chapters, we have introduced the domains of the agents, the believable agents, the chatterbots and the insertion of personality in a program. The different tools and information needed to develop our chatterbot have been tackled in each of these domains. Using these information and tools, we have designed and implemented the chatterbot explained in the fourth chapter. This work, done at UTS (University of Technology, Sydney), is still in progress and has already given good results. Indeed, by giving the chatterbot richly provided mind sets (XML files), it can already talk like a human on some themes. A user, ignoring he talks with a machine, can have a short conversation with our chatterbot without guessing he is actually talking to a program. This good result allows us to think that the architecture chosen for the design of the bot is good. Another encouraging point is the fact that the chatterbot has been designed to allow further modifications and add-ons.

Our belief is that, by giving the program some add-ons like learning or pattern's usage, the chatterbot couldn't be differentiated from a human by a normal chatter (not a member of the Loebner Contest's jury).

There is still a lot of work to provide in this area of research, but we believe these researches will soon lead to results that could revolutionize the virtual worlds.

Appendices

I. Documentation of the program

1. How to run the program

The first thing to do is to launch the batch file named *Chatterbot.bat*. Then, a fictive chat room interface appears (public chat). This interface represents the Stock Exchange's chat room in which we can write some sentences in order to test and simulate the "bad adviser" person.

In order to do that, we first have to write a name followed by a space and double point (example: "Bob :"). The name of a person has to be different each time the program is tested. Then, whatever sentence can be written but only sentences with "bad advice(s)" will be detected. When it is done, another frame named "private chat" appears. This frame is only there to see the Agent to "bad adviser" conversation (in fact, as the B.A.D BOT program is able to launch as many agents as "bad adviser" detected, there will also be as many "private chat" frames as "bad adviser" detected).

2. Parameters

- Double clusterlimit = 0.4;

This parameter consists of the minimum clusterweight between two words in order to be taken into account in the computations. Depending on the situations, 03 to 1.2 may be used.

- Boolean assymetric_clustering = **true**;

with asymmetric clustering, f(word1,word2) <> f(word2,word1)
if this parameter is false:
g(word1,word2) = (f(word1,word2)+f(word2,word1))/2

- String explore_cooc = "DEPENDANT"; // or INDEPENDENT

In an independent exploration of a sentence, each word is taken as a separate entity and the final two words that are chosen are simply the two that achieve the highest cluster weight together.

In a dependant exploration, the sentence is seen as an entity. This method is more complex but gives more accurate results.

- **int** headlineprecision = 2;

It defines the minimum amount of times that 2 words have to be present in the same news headline.

- String type_keybord = "QWERTY"; // or AZERTY

It defines the type of keyboard the chatterbot "uses" in order to introduce typing errors accordingly

- **int** maxTimesWithBot = 10;

The maximum amount of times the bot can speak to the same person.
Note that the bot never speaks to the same person twice the same day.

- **int** minWeightToLaunch = 25;

The minimum bogusweight that is necessary before the bot begins a conversation.

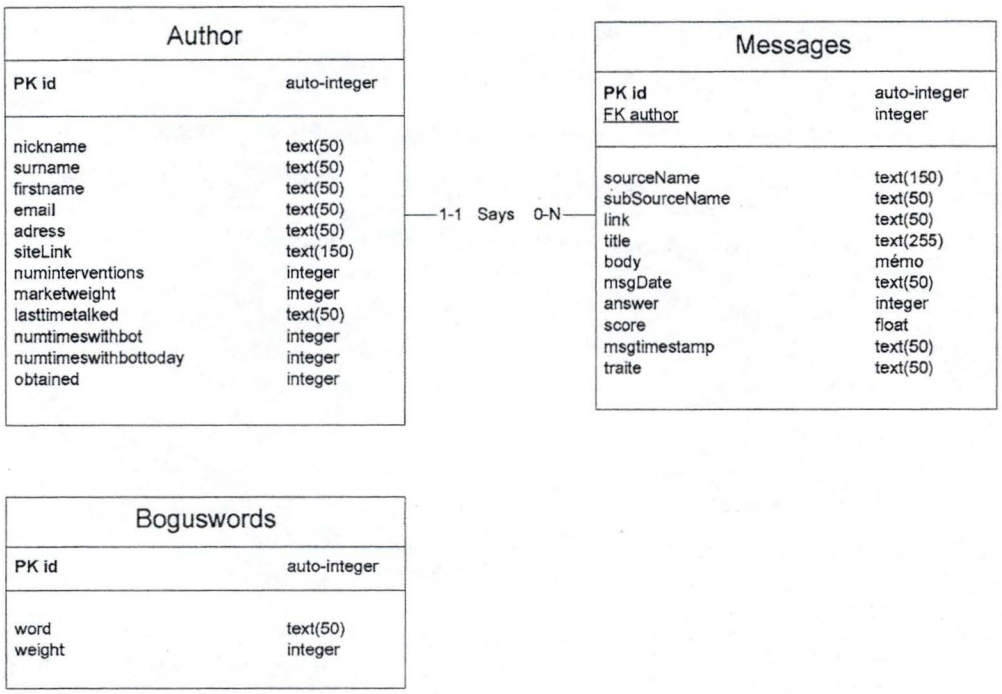
3. DataBases

1. Design of the databases and accesses within the program

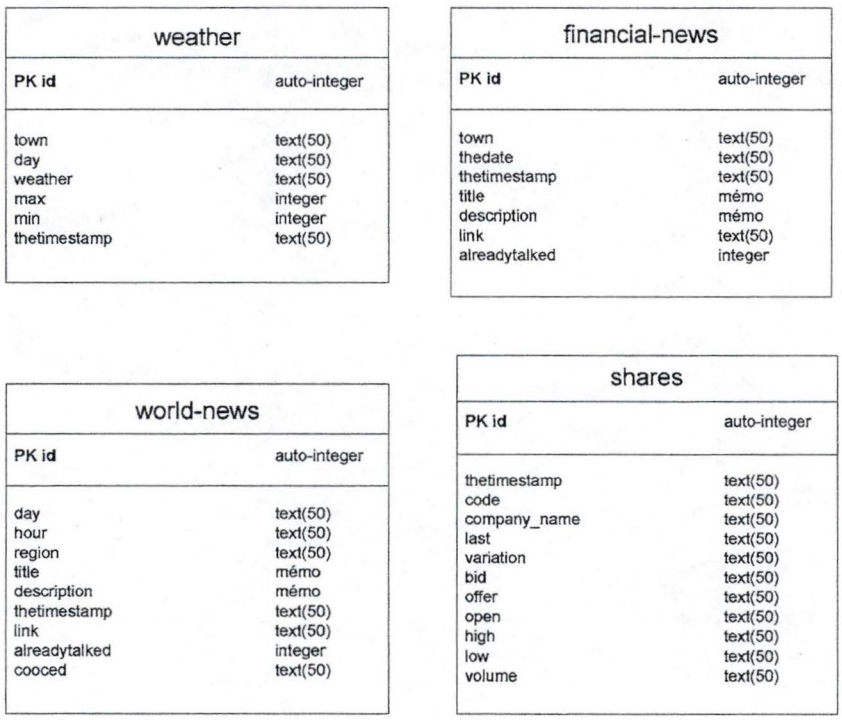
Package: Acces

Classes: Bd, Bd2

Adapted Smarts DataBase, still compatible: managed in class Bd



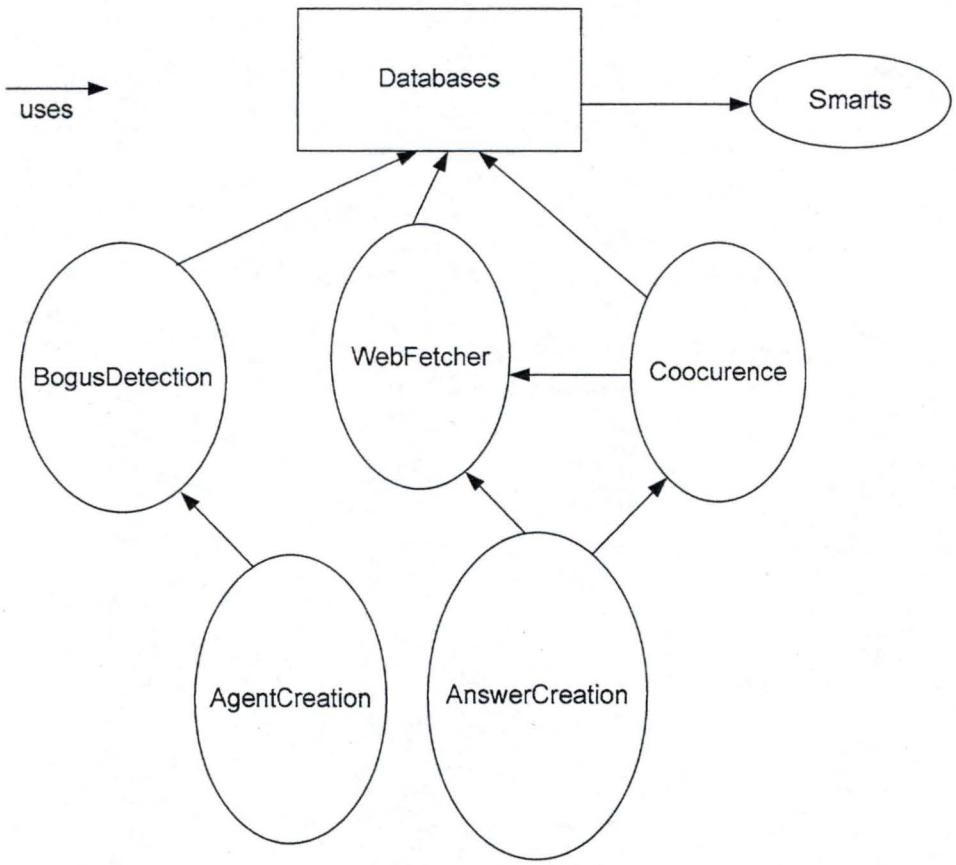
Bdknowledge DataBase: managed in class Bd2



The database access methods are centralised in one class per DataBase, where the concurrent accesses are treated. This centralisation allows the programmers to delegate the treatment of exceptions to these classes.

The program works on two separate databases in order to avoid unnecessary waiting times if a thread accesses a table from the database that is not even linked to an already blocked one.

4. High level design



5. Detail design

Bogus Detection

Vocabulary

Bogus advice: false advice on buying shares in order to manipulate prices and persons
Share: financial part of a company
Sharecode: each share is identified by a 3, 4 or 5 letter code

Design

Package : Bogus & General
Classes : Extrac, Lemain, ShareKnowledge, Boguswords, StartBug, General (followFromSmarts)

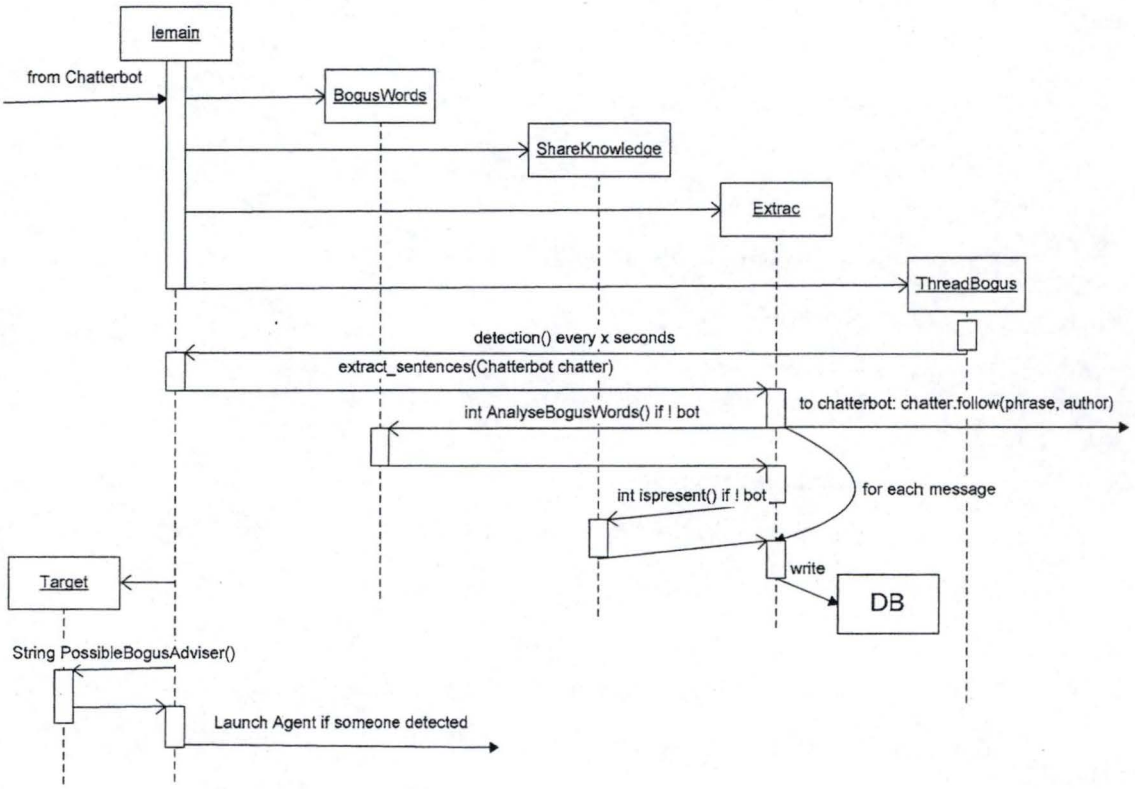
As opposed to the coocurence analysis which is a more complex part, the detection has been deliberately kept simple and fully understandable for the user of Bad Bots. The purpose is that the person knows and controls who is picked up by our program. The detection of a bogus adviser is based on keywords chosen by the user as well as a "stay in topic" detector. This means that a person speaking about any share will be more easily chosen than a person speaking about whatever else.

Choosing the person the talk to

Package: Bogus
Class: Target

Once a person on the chat has been identified as (perhaps) someone who gives away bogus advices, an agent is launched to talk to him. The agent will only be launched if the detected person is not an agent itself and if the person talked

within the last 5 minutes. The agent is launched if the bot didn't speak to him during the present day and not more then a maximum amount of times in total. (See also the section about the parameters)



Bogus Adapt

Application : bogusAdapt
Classes : AccesDB, bogusFrame, newBogusFrame, General

Separate application that allows the user to add/modify/delete the keywords to be detected as being an indication for bogus as well as their weight (level of importance in the detection).

Keywords are ordered alphabetically and the user can easily keep an eye on all modifications done within the session. There is an integrated control to avoid doublets and subsequence of words.

Agent creation

System design

If the avatars are to be "believable" then their design must have an unifying conceptual basis. This is provided here by the agent's *character*. The first decision that is made when an agent is created is to select its character using a semi-random process that ensures that multiple instances of the agent in close virtual proximity have identifiably different characters.

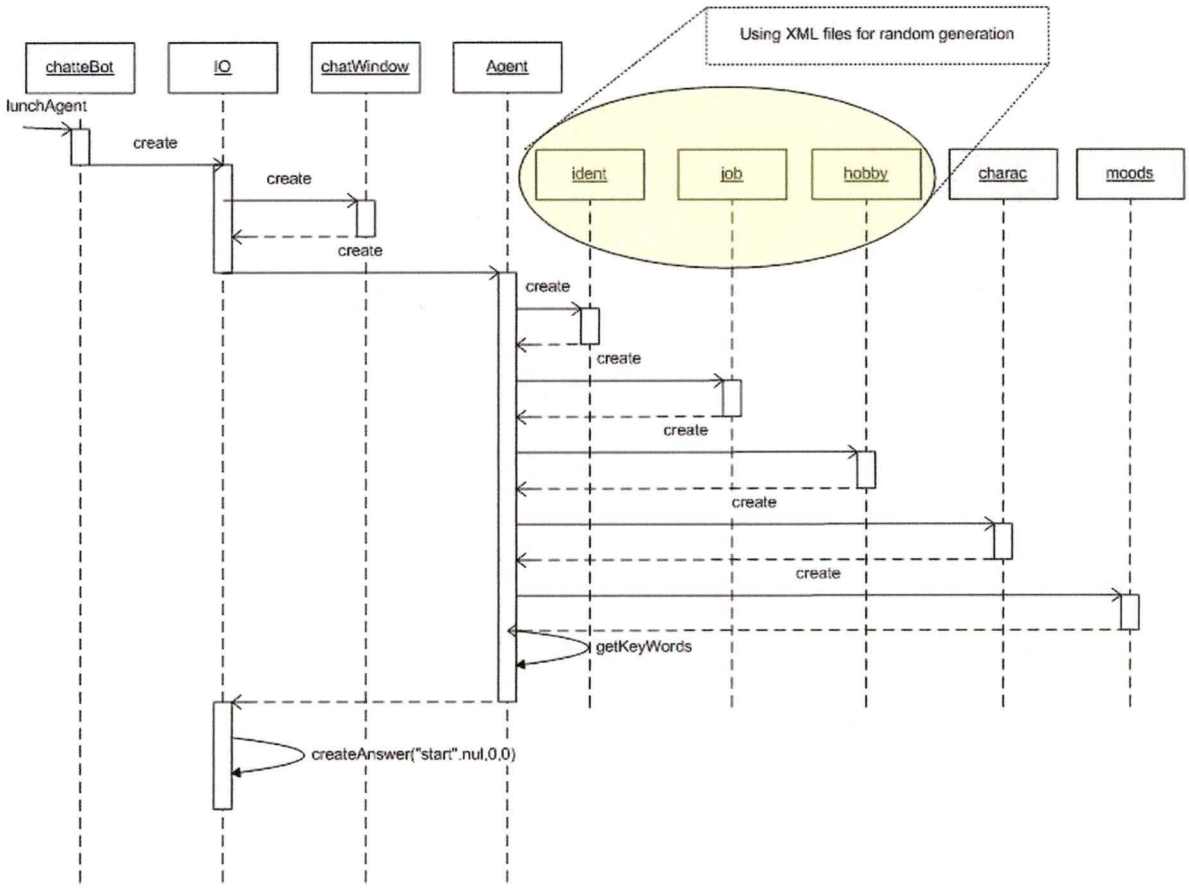
The dimensions of character that we have selected are intended specifically for a finance-based environment.

- Politesse means the use of polite words, phrases and forms
- Dynamism is the tendency to react rapidly, succinctly and vigorously
- Optimism here means a tendency to use up-beat phrases and the tendency to not use negations
- Self-confidence here means the tendency to respond with declarative statements rather than tentative propositions or questions.

The selection of an agent's character does not alone determine its behaviour. Each agent's behaviour is further determined by its moods that vary slowly. The dimensions of moods that we have identified are:

- Happiness
- Sympathy

Implementation design



Answer creation

System design

The generated answer has to be consistent with the personality and the mood of the character and if possible, the answer will contain some news from Internet.

In order to achieve that, XML technology is used for response's creation. In fact, three XML files are used: **listWords.xml**, **ListAnsWithoutOnt.xml**, **ListAnswerOnt.xml**.

First, the sentence which comes from the chat room is parsed in order to find a topic for the answer's creation. There are five different kind of possible subjects: Shares, World News, Weather, Financial News and "no subject".

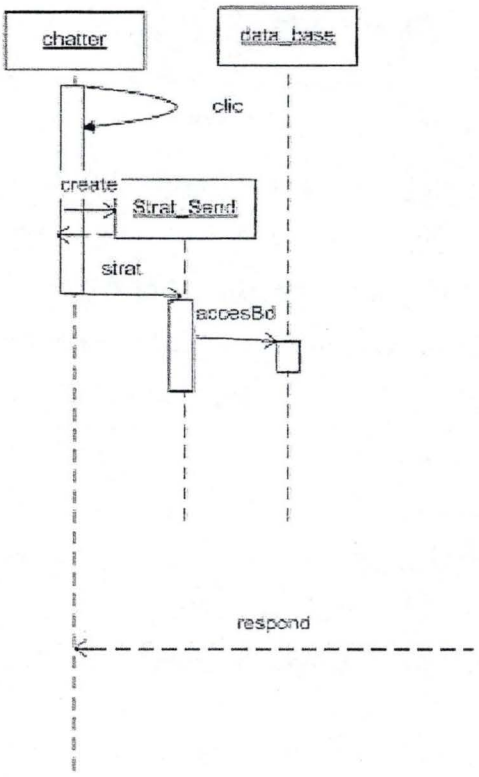
At this point, **ListWords** is use to find a "thema" corresponding to a given list of words (topic). The obtained "thema" allows now to choose a focus node (in one of the too others XML files) having the same attribute "thema". ("what's your name?" or "what's your job?" or something else!). We have now two possibilities: the found topic is "no subject" or a thema (with a high priority) has been found then we are going to use the **ListAnsWithoutOnt** file; on the other cases, we are going to use the **ListAnswerOnt** file.The attribute markup from the markupPhrases node represents, for the first figure, the politeness, and for the second, the optimism of an Agent. It allows then to select an output sentence corresponding to the personality of an Agent.

A way to look more human is to introduce random keyboard mistakes into the chatterbot's replies. The choice is left to the user of our program to have the chatterbot using a "qwerty" or "azerty" keyboard, as the mistakes are different on both types of keyboards.

Implementation design

First step: The "fictive" chat room's launching.

The final version of this is achieved by the S.M.A.R.T program. This one fetches the sentences on the chat in order to store them in the database.

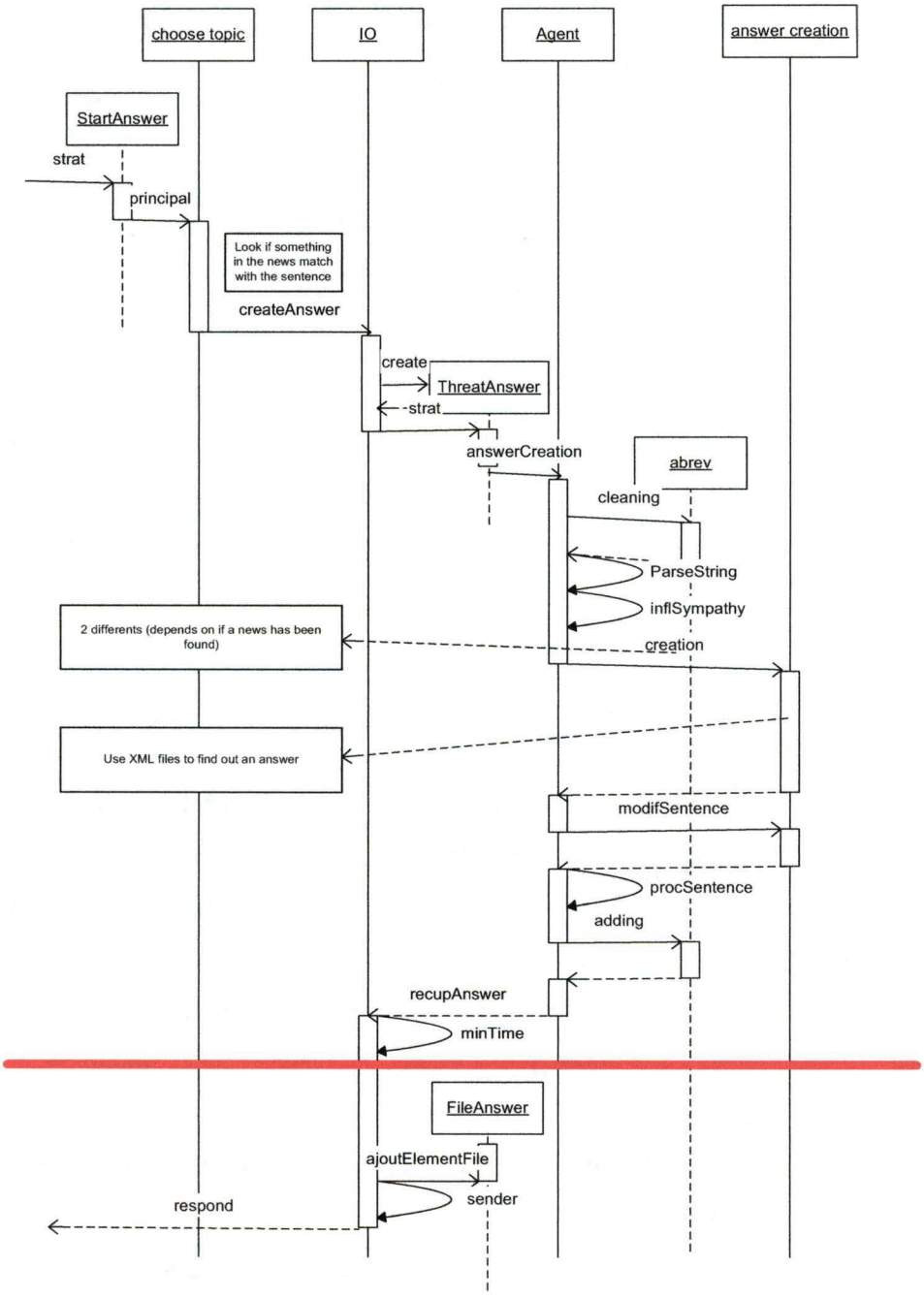


Second step: the method "followFromSmart" which is called by the bogus detection module launches every **x** second(s) a thread named StartAnswer per sentences that have to be answered.

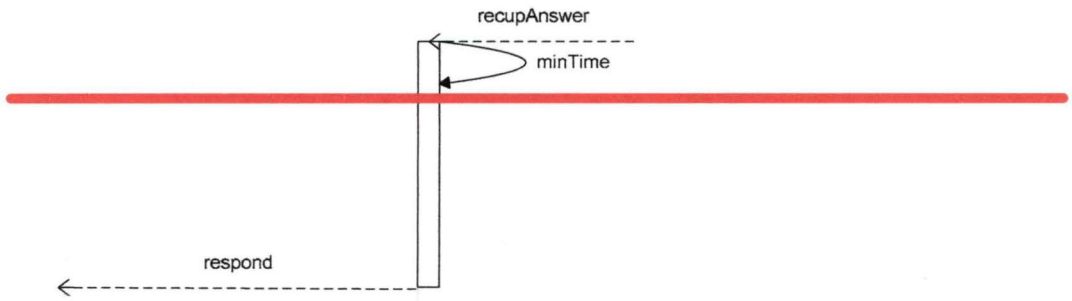
After generating the answer, there are two possibilities:

If there are two (or more) StartAnswer threads launched before one has finished its execution (it is the case when several sentences from a same user are waiting for an answer), then the others threads are placed in a waiting line named "FileAnswer"

Else (only one sentence has to be answered), the method "respond" is directly launched.



In the case of only one sentence to be answered: the scheme is the same except for the end. So, the end for this second case is given in the next figure.



Coocurence and Performance

Vocabulary

Clustering: grouping of objects based on similarity, dissimilarity or other any other property

Coocurence: clustering method based on occurrences of words in the same texts

Package: Coocurence

Class : Cooc

The initial model we chose to use is a clustering method that calculates a weight between words representing their coocurence level in a series of texts. Unfortunately, in spite of very nice suggestions obtained with the method, we had to adapt it to reply with a particular newsheadline (full sentence), which actually gives results that are less impressive, but still accurate if the program is well parametered. Each phrase a user says can be analyzed by the bot in a dependant or independent way.

A particular attention has been brought to optimization, both in memory and real-time performance.

The clustering method is asymmetric; this means that the weight between word1 and word2 is not the same as between word2 and word1.

We deliver one light, optimized, clean and tested version of the Cooc class (the Cooc class alone) without concepts and one for test purposes which finds concepts and takes them into account but computes redundant information and

considerably multiplies the execution time and memory usage. The gain it gives to the chatterbot is unknown.

The Cooc method works in three phases.

1) A few times a day: building a datastructure representing the content of the texts as well as mathematical computations. Some concessions have been made in this phase to improve the speed of the next real-time phases. This happens every time news has successfully been fetched from the internet. (x times a day, nothing is computed is called if no news has been retrieved)

If concepts are not used, this part has been brought to 10 seconds in its actual version thanks to optimization techniques. (for 100 texts and 15000 words on a Celeron 1000 with 256 MB Ram).

2) for every intervention of the chattermate: word to word clustering

3) choice of the best newsheadline

The concept network

What is a concept?

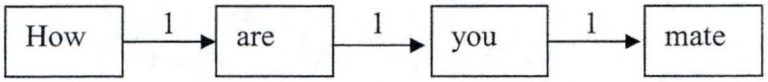
A concept can be a one-, two-, three, or more-word phrase. If a word is almost every time followed by another one, we can say that it is a concept.

Prime minister, hunger strike, George W Bush, foreign affaires, ... are examples of concepts.

How to find a concept?

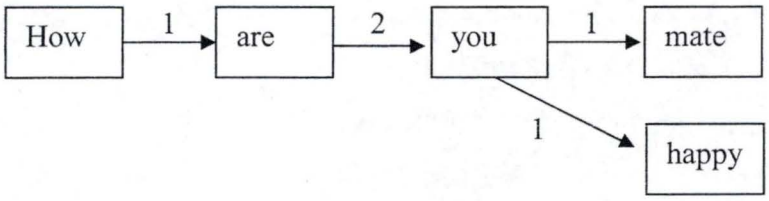
The method used is to build a network. We analyse a text and put a link between each word which follows another one. For example if we have this text "How are you mate?", there will be a link between "How" and "are", a link between "are" and "you", and a link between "you" and "mate".

We can represent the network graphically:



A network is composed by nodes and relations. "How", "are", "you", and "mate" are the nodes. For the moment there is a weigh of 1 for each relation between the nodes.

Now let's update the network with a new sentence: "Are you happy?"



We see that now there is a weight of two between "are" and "you". We can continue and at the end, the nodes which have a big relationship are perhaps concepts.

Actually, we say that two words are a concept if they are more often together than with another word. That means that we have to remember for each word how many times it appears at all. A word has to appear 3/4 of its total appearances with another to be a concept.
For example if Tony appears 100 times at all, it has to appear at least 75 times with Blair to be a concept.

Some precisions:

In order to penalize general terms (terms which appear in many places), we make a list of common words. These words will not be including in any concepts. General terms are words like you, will, would, can, make, do, … Between words with capital letters, we don't add a weight of 1 but a weight of 3. It is because words with capital letters have more chance to be part of a concept.

"Blair witch" is a concept. "Tony Blair" is a concept too. But "Tony Blair witch" isn't a concept. So there is a detection to see if "witch" and "Tony" appear almost the same number of time. There is a 20% limit allowed. For example if Tony appear 10 times and Blair appear just behind 8 times, than it is a concept. If Tony appears 10 times, and witch appears 7 times, it will not be a concept.

How to represent the network?

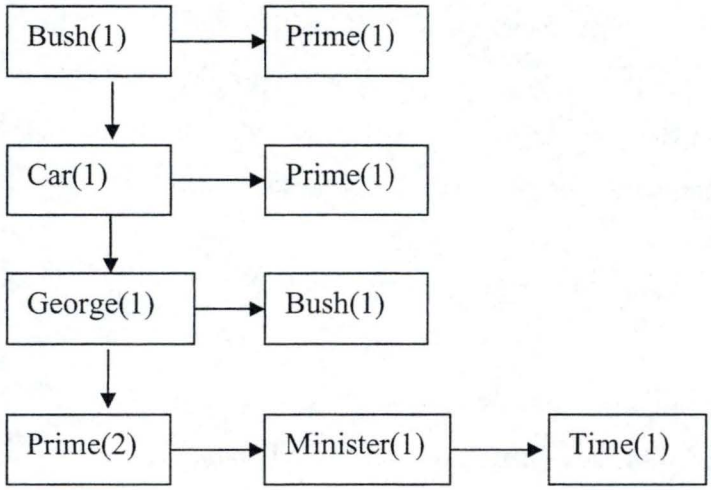
The most common way to represent a network is to use a matrix.
For example the network build above can be representing like this:

	how	are	you	mate	happy
how	0	**1**	0	0	0
are	0	0	**2**	0	0
you	0	0	0	**1**	**1**
mate	0	0	0	0	0
happy	0	0	0	0	0

We can see that there are a lot of zeros in this structure. So in order to optimize the representation, we choose to make a linked list which contains only the not-zero cases of the matrix.

Here is the exact structure of the linked list with this two sentences:

- George Bush Prime Minister
- Car Prime time



The first column (the left one) represents what we call "words" and the other column (here there are only 2 others) represent what we call "links". The number between brackets has a different signification for the words and for the links. For the words it represents the total number of occurrences of this word. For the links it represents the number of time that the link appears just behind the word of row. For example "Prime" appears two times in all the texts, and "Time" appears one time behind "Prime".

The News Fetcher

The person that we are going to interact with is the person that talks a lot about finance and particularly, Australian shares.
Where will we take the information from?

Shares

The link to follow is from the official Australian Stock exchange website, the access to the information is quiet complicated because there is no direct link to each share. The website uses a combobox and we have to select all the shares one by one.

- http://www.asx.com.au/asx/markets/EquitySearchPage.jsp?template=sf11000&issuername=&x=7&y=8

Financial News

- Australia & NZ Finance (http://au.dailynews.yahoo.com/finance/reutersfinance/)
  - Headline
  - Location
  - Month (3 letters) comma day (digits)

World News

ABC news online seems the best website, the information is structured and complete. We save the headlines together with the date of the news and the full story associated. This information are also used to build the co-occurrences structure and the concept network.

- ABC news online http://www.abc.net.au/news/justin/default.htm

Weather

Also taken from ABC, a 4 day weather forecast with a very short comment for each day. Information is available for most Australian cities.

- Australia Centres Weather Forecast (http://www.abc.net.au/news/australia/weather/default.htm)

Purgatoire

Package : Purger

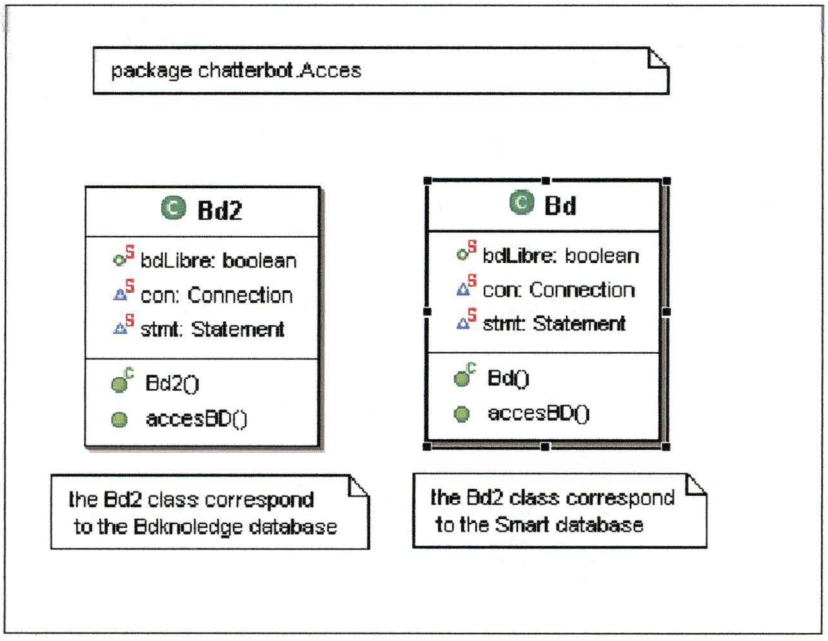
Class : Purgatoire

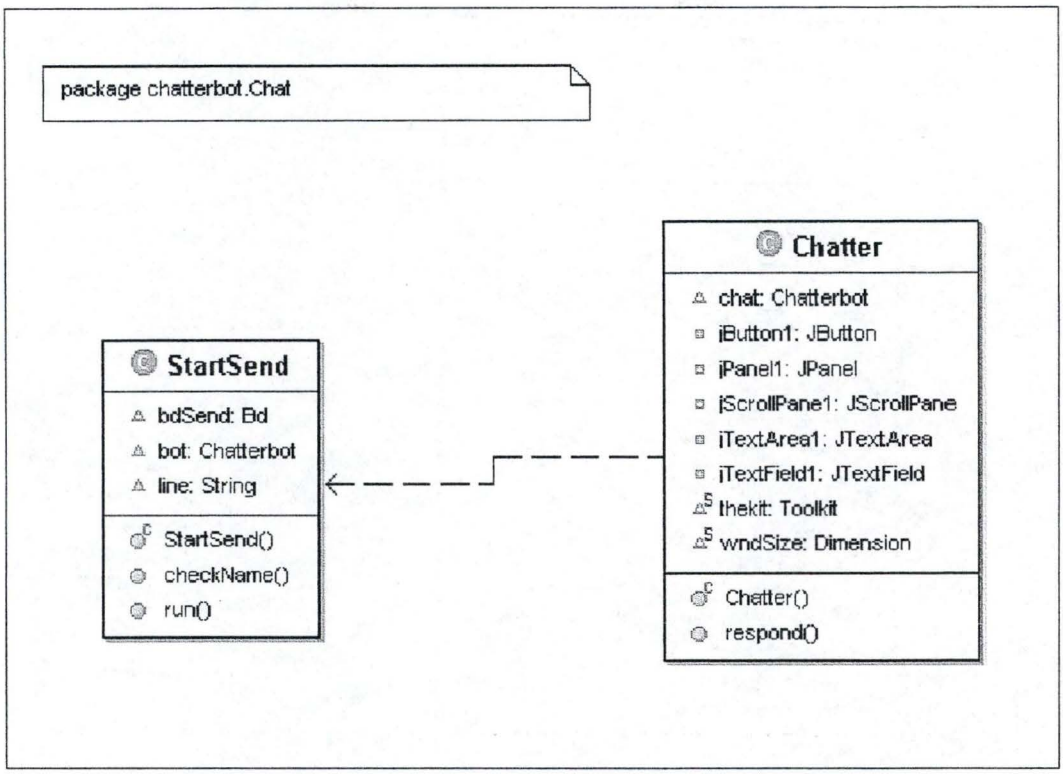
The chatterbot has to be able to acquire new knowledge and to stay up-to-date with its environment but it would be silly if he talked suddenly about a breaking news that happened 3 weeks ago …

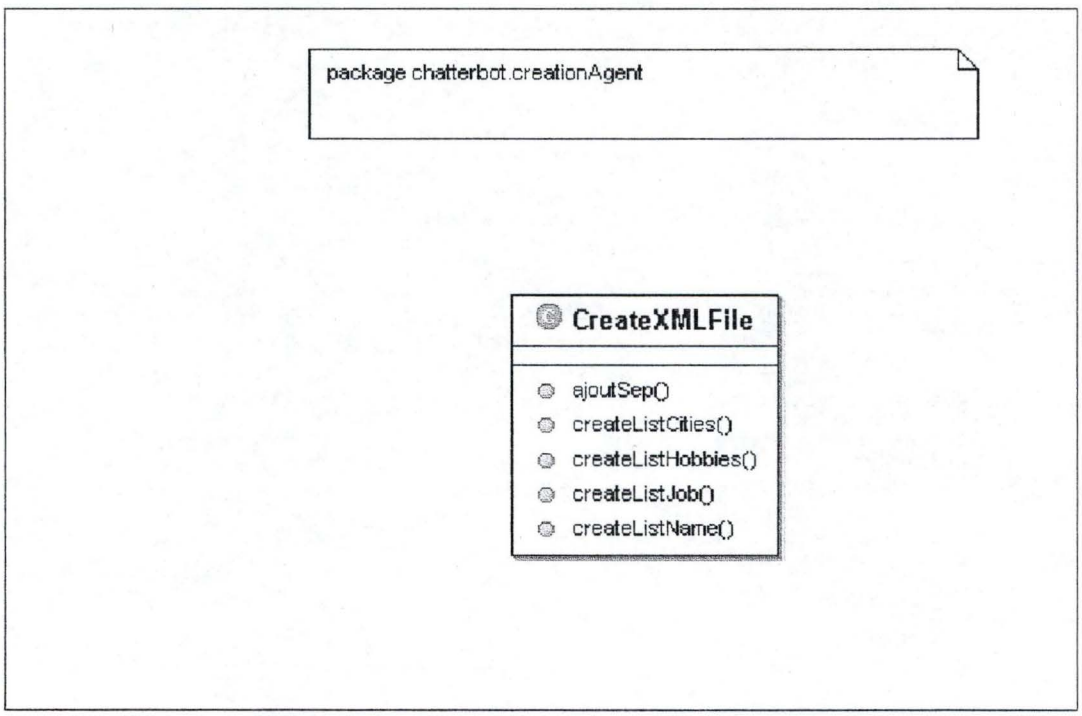
The Purger is called once a day and has a forgetting curve that is not linear, as the bot wipes his knowledge of, based on a probability and the age of the knowledge.

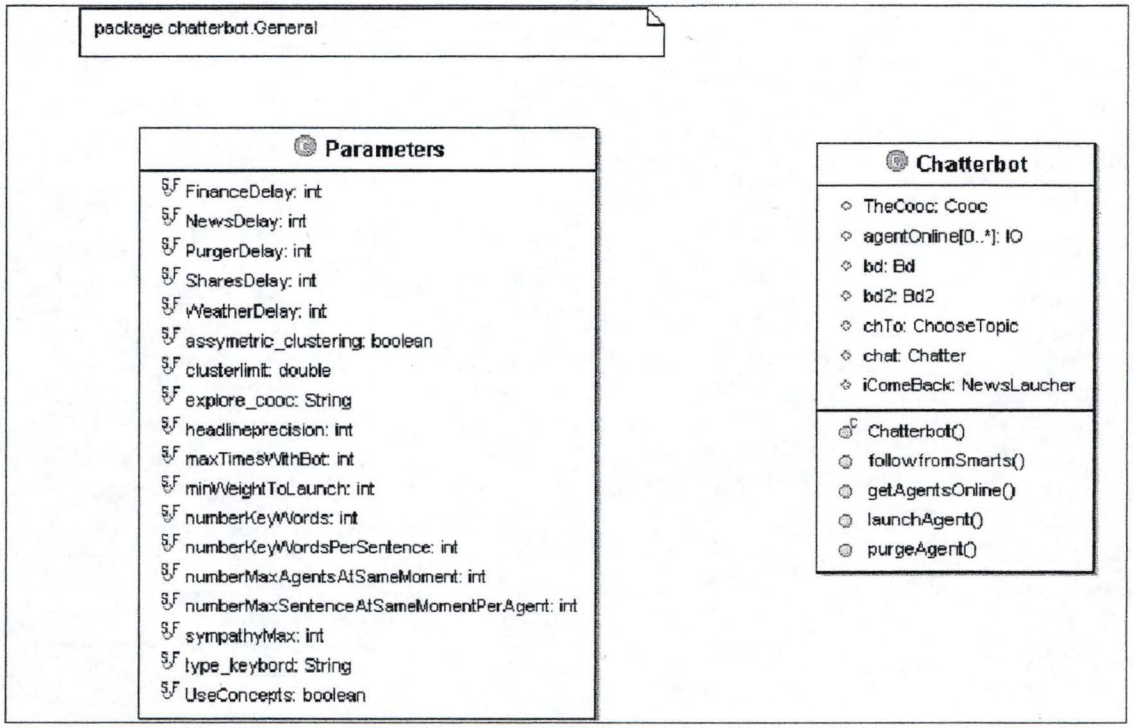
Weather and shares don't have to be purged, because the content is updated and not added.

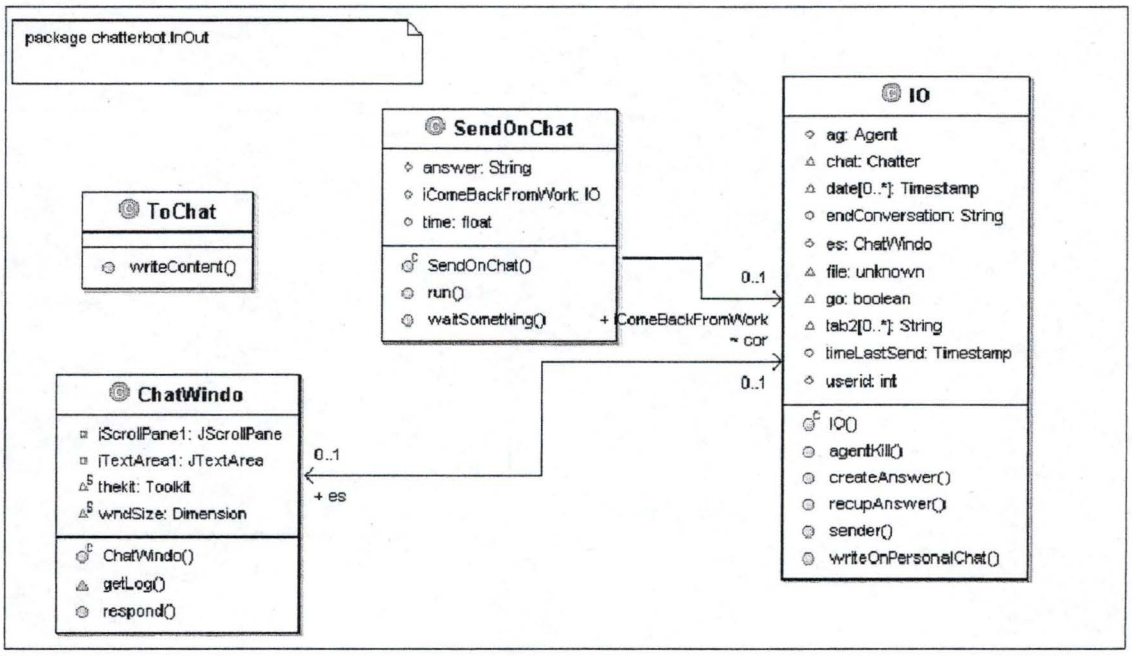
6. Class Diagrams

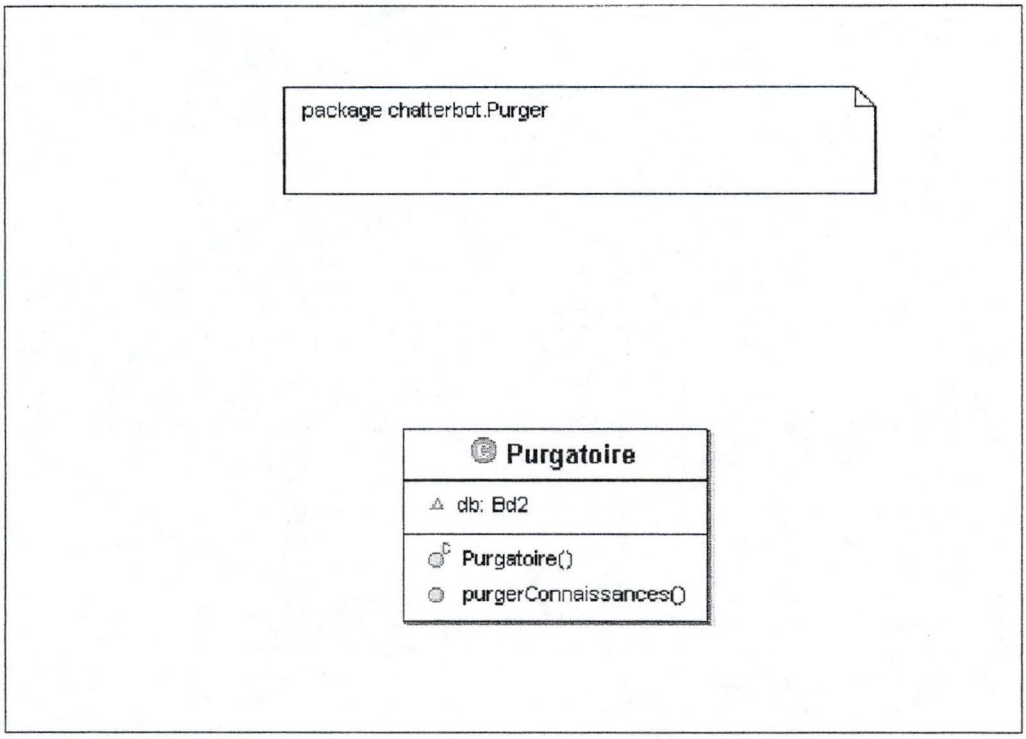


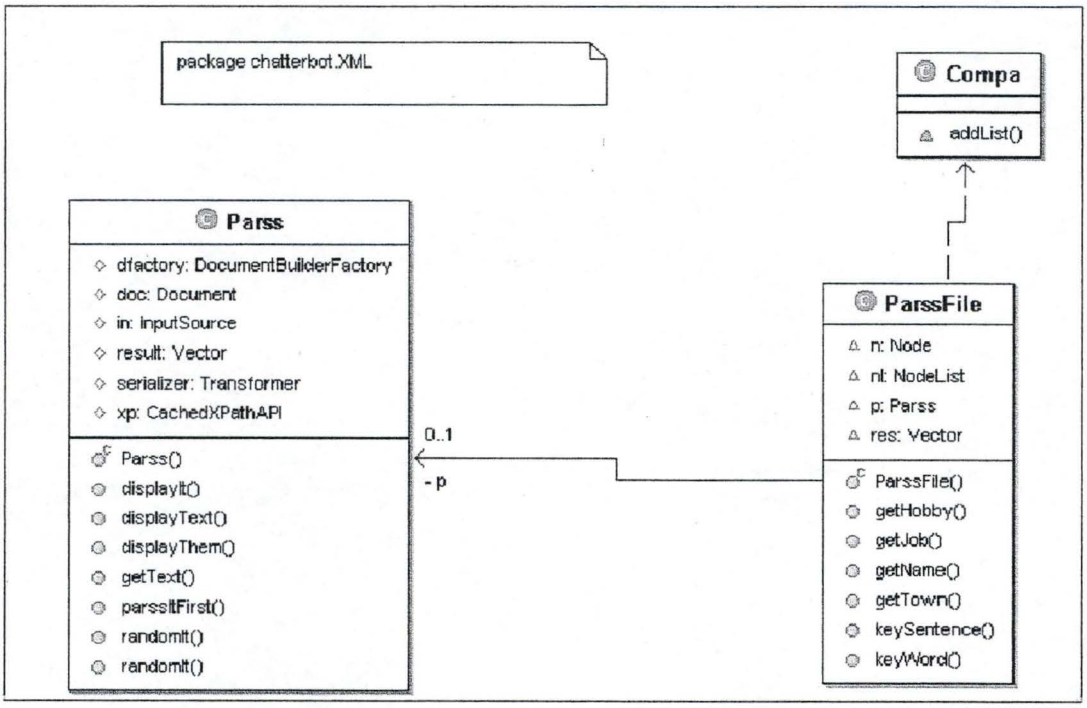

package chatterbot.Chat
Chatter
chat: Chatterbot
jButton1: JButton
jPanel1: JPanel
jScrollPane1: JScrollPane
jTextArea1: JTextArea
jTextField1: JTextField
thekit: Toolkit
wndSize: Dimension
Chatter()
respond()
StartSend
bdSend: Bd
bot: Chatterbot
line: String
StartSend()
checkName()
run()

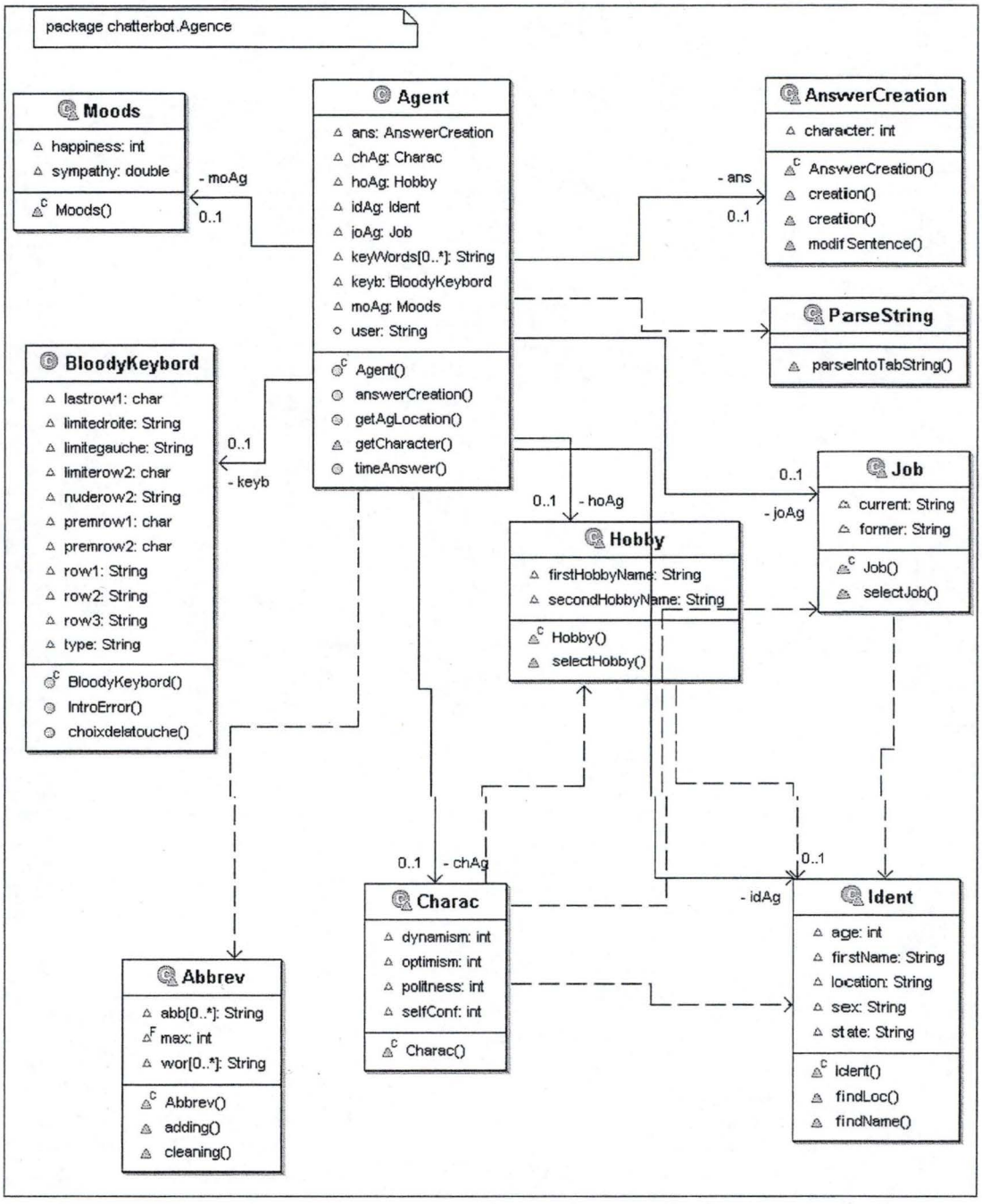

package chatterbot.creationAgent
CreateXMLFile
ajoutSep()
createListCities()
createListHobbies()
createListJob()
createListName()

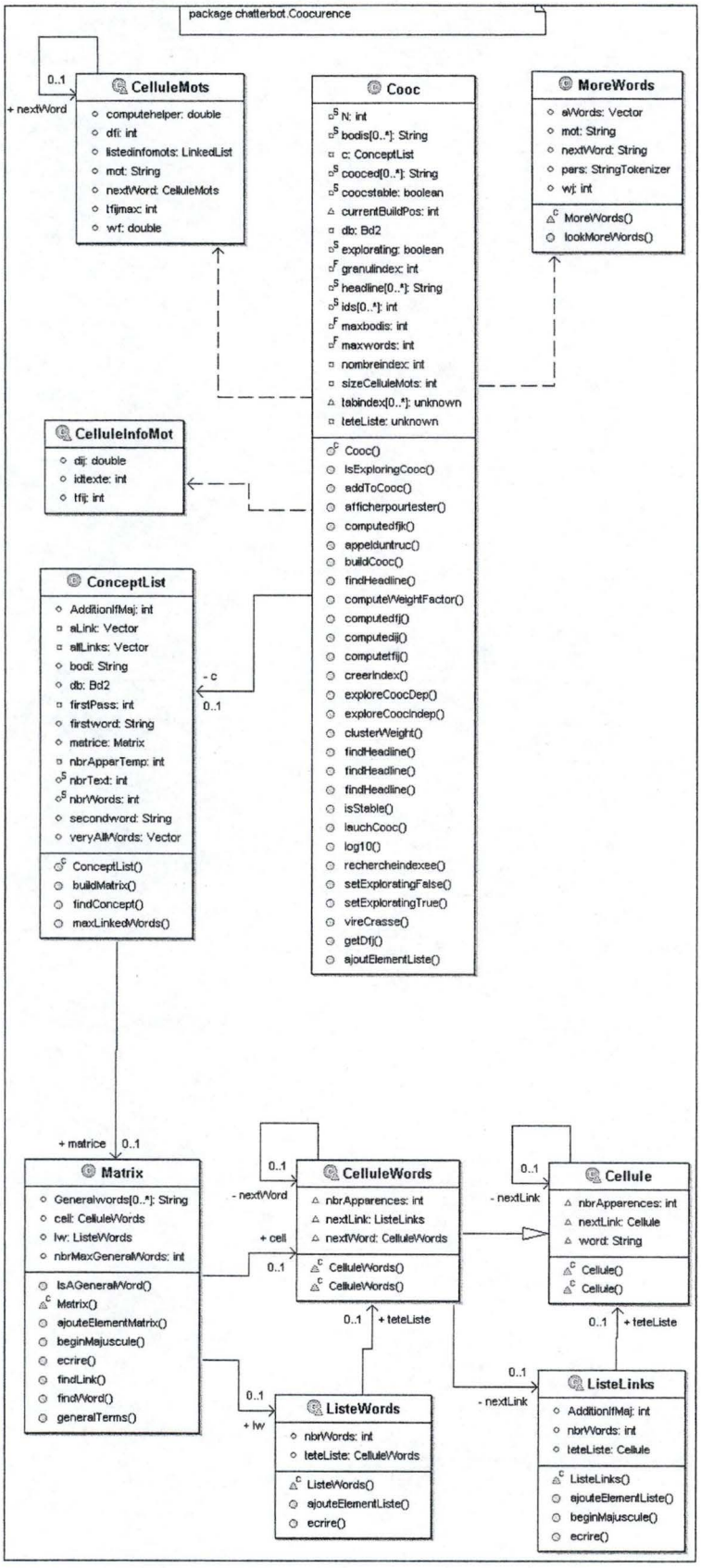

package chatterbot.General
Parameters
FinanceDelay: int
NewsDelay: int
PurgerDelay: int
SharesDelay: int
WeatherDelay: int
assymetric_clustering: boolean
clusterlimit: double
explore_cooc: String
headlineprecision: int
maxTimesWithBot: int
minWeightToLaunch: int
numberKeyWords: int
numberKeyWordsPerSentence: int
numberMaxAgentsAtSameMoment: int
numberMaxSentenceAtSameMomentPerAgent: int
sympathyMax: int
type_keybord: String
UseConcepts: boolean
Chatterbot
TheCooc: Cooc
agentOnline[0..*]: IO
bd: Bd
bd2: Bd2
chTo: ChooseTopic
chat: Chatter
iComeBack: NewsLaucher
Chatterbot()
followfromSmarts()
getAgentsOnline()
launchAgent()
purgeAgent()

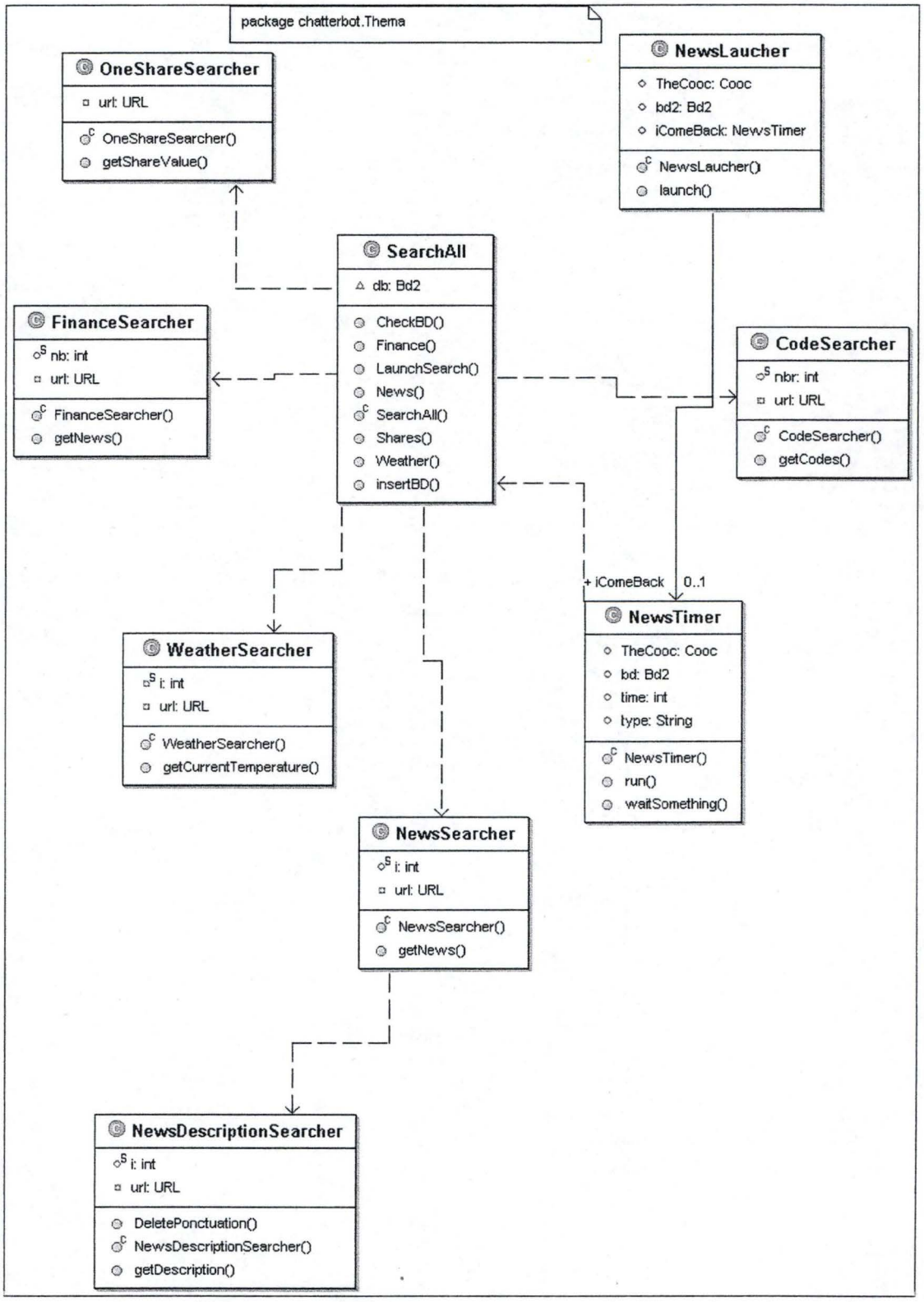

package chatterbot.InOut
ToChat
writeContent()
SendOnChat
answer: String
iComeBackFromWork: IO
time: float
SendOnChat()
run()
waitSomething()
0..1
+ iComeBackFromWork
~ cor
0..1
IO
ag: Agent
chat: Chatter
date[0..*]: Timestamp
endConversation: String
es: ChatWindo
file: unknown
go: boolean
tab2[0..*]: String
timeLastSend: Timestamp
userid: int
IO()
agentKill()
createAnswer()
recupAnswer()
sender()
writeOnPersonalChat()
ChatWindo
jScrollPane1: JScrollPane
jTextArea1: JTextArea
thekit: Toolkit
wndSize: Dimension
ChatWindo()
getLog()
respond()
0..1
+ es

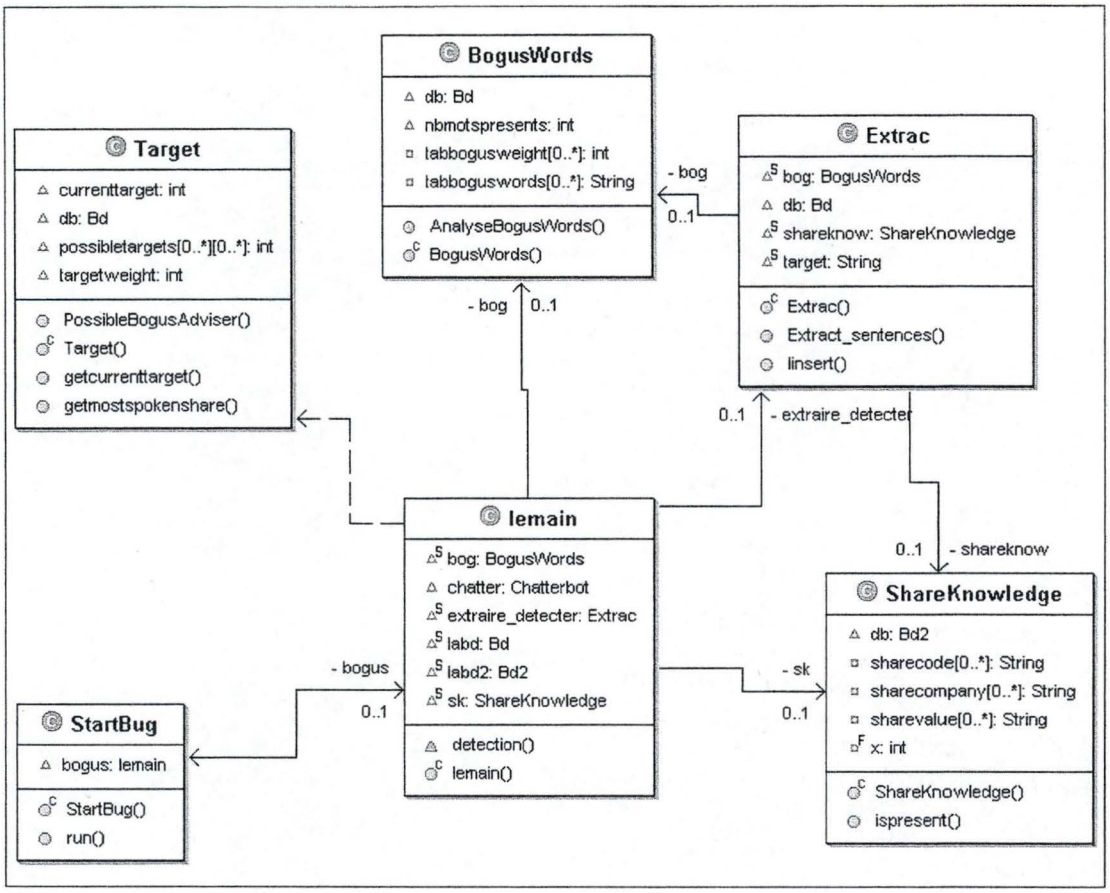

package chatterbot.Purger
Purgatoire
db: Bd2
Purgatoire()
purgerConnaissances()


package chatterbot.XML
Compa
addList()
Parss
dfactory: DocumentBuilderFactory
doc: Document
in: InputSource
result: Vector
serializer: Transformer
xp: CachedXPathAPI
Parss()
displayIt()
displayText()
displayThem()
getText()
parssItFirst()
randomIt()
randomIt()
0..1
- p
ParssFile
n: Node
nl: NodeList
p: Parss
res: Vector
ParssFile()
getHobby()
getJob()
getName()
getTown()
keySentence()
keyWord()


package chatterbot.Agence
Moods
happiness: int
sympathy: double
Moods()
- moAg
0..1
Agent
ans: AnswerCreation
chAg: Charac
hoAg: Hobby
idAg: Ident
joAg: Job
keyWords[0..*]: String
keyb: BloodyKeybord
moAg: Moods
user: String
Agent()
answerCreation()
getAgLocation()
getCharacter()
timeAnswer()
- ans
0..1
AnswerCreation
character: int
AnswerCreation()
creation()
creation()
modifSentence()
ParseString
parseIntoTabString()
BloodyKeybord
lastrow1: char
limitedroite: String
limitegauche: String
limiterow2: char
nuderow2: String
premrow1: char
premrow2: char
row1: String
row2: String
row3: String
type: String
BloodyKeybord()
IntroError()
choixdeletouche()
0..1
- keyb
0..1
- hoAg
0..1
- joAg
Job
current: String
former: String
Job()
selectJob()
Hobby
firstHobbyName: String
secondHobbyName: String
Hobby()
selectHobby()
0..1
- chAg
0..1
- idAg
Charac
dynamism: int
optimism: int
politness: int
selfConf: int
Charac()
Ident
age: int
firstName: String
location: String
sex: String
state: String
Ident()
findLoc()
findName()
Abbrev
abb[0..*]: String
max: int
wor[0..*]: String
Abbrev()
adding()
cleaning()


package chatterbot.Coocurence
CelluleMots
computehelper: double
dfi: int
listedinfomots: LinkedList
mot: String
nextWord: CelluleMots
tfijmax: int
wf: double
0..1
+ nextWord
Cooc
N: int
bodis[0..*]: String
c: ConceptList
cooced[0..*]: String
coocstable: boolean
currentBuildPos: int
db: Bd2
exploratingi: boolean
granulindex: int
headline[0..*]: String
ids[0..*]: int
maxbodis: int
maxwords: int
nombreindex: int
sizeCelluleMots: int
tabindex[0..*]: unknown
teteListe: unknown
Cooc()
IsExploringCooc()
addToCooc()
afficherpourtester()
computedfjk()
appelduntruc()
buildCooc()
findHeadline()
computeWeightFactor()
computedfj()
computedij()
computetfij()
creerIndex()
exploreCoocDep()
exploreCoocIndep()
clusterWeight()
findHeadline()
findHeadline()
findHeadline()
isStable()
lauchCooc()
log10()
rechercheindexee()
setExploratingFalse()
setExploratingTrue()
vireCrasse()
getDfj()
ajoutElementListe()
MoreWords
alWords: Vector
mot: String
nextWord: String
pars: StringTokenizer
wj: int
MoreWords()
lookMoreWords()
CelluleInfoMot
dij: double
idtexte: int
tfij: int
ConceptList
AdditionIfMaj: int
aLink: Vector
allLinks: Vector
bodi: String
db: Bd2
firstPass: int
firstword: String
matrice: Matrix
nbrApperTemp: int
nbrText: int
nbrWords: int
secondword: String
veryAllWords: Vector
ConceptList()
buildMatrix()
findConcept()
maxLinkedWords()
- c
0..1
+ matrice
0..1
Matrix
GeneralWords[0..*]: String
cell: CelluleWords
lw: ListeWords
nbrMaxGeneralWords: int
IsAGeneralWord()
Matrix()
ajouteElementMatrix()
beginMajuscule()
ecrire()
findLink()
findWord()
generalTerms()
0..1
- nextWord
CelluleWords
nbrApperences: int
nextLink: ListeLinks
nextWord: CelluleWords
CelluleWords()
CelluleWords()
+ cell
0..1
0..1
- nextLink
Cellule
nbrApperences: int
nextLink: Cellule
word: String
Cellule()
Cellule()
0..1
+ teteListe
0..1
+ teteListe
0..1
- nextLink
ListeLinks
AdditionIfMaj: int
nbrWords: int
teteListe: Cellule
ListeLinks()
ajouteElementListe()
beginMajuscule()
ecrire()
0..1
+ lw
ListeWords
nbrWords: int
teteListe: CelluleWords
ListeWords()
ajouteElementListe()
ecrire()


package chatterbot.Thema
OneShareSearcher
url: URL
OneShareSearcher()
getShareValue()
NewsLaucher
TheCooc: Cooc
bd2: Bd2
iComeBack: NewsTimer
NewsLaucher()
launch()
SearchAll
db: Bd2
CheckBD()
Finance()
LaunchSearch()
News()
SearchAll()
Shares()
Weather()
insertBD()
FinanceSearcher
nb: int
url: URL
FinanceSearcher()
getNews()
CodeSearcher
nbr: int
url: URL
CodeSearcher()
getCodes()
+ iComeBack
0..1
NewsTimer
TheCooc: Cooc
bd: Bd2
time: int
type: String
NewsTimer()
run()
waitSomething()
WeatherSearcher
i: int
url: URL
WeatherSearcher()
getCurrentTemperature()
NewsSearcher
i: int
url: URL
NewsSearcher()
getNews()
NewsDescriptionSearcher
i: int
url: URL
DeletePonctuation()
NewsDescriptionSearcher()
getDescription()


BogusWords
db: Bd
nbmotspresents: int
tabbogusweight[0..*]: int
tabboguswords[0..*]: String
AnalyseBogusWords()
BogusWords()
Target
currenttarget: int
db: Bd
possibletargets[0..*][0..*]: int
targetweight: int
PossibleBogusAdviser()
Target()
getcurrenttarget()
getmostspokenshare()
Extrac
bog: BogusWords
db: Bd
shareknow: ShareKnowledge
target: String
Extrac()
Extract_sentences()
linsert()
- bog
0..1
- bog
0..1
0..1
- extraire_detecter
0..1
- shareknow
lemain
bog: BogusWords
chatter: Chatterbot
extraire_detecter: Extrac
labd: Bd
labd2: Bd2
sk: ShareKnowledge
detection()
lemain()
- bogus
0..1
- sk
0..1
StartBug
bogus: lemain
StartBug()
run()
ShareKnowledge
db: Bd2
sharecode[0..*]: String
sharecompany[0..*]: String
sharevalue[0..*]: String
x: int
ShareKnowledge()
ispresent()

II. Examples of the mind sets

Some examples of the mind sets (XML files) will be provided in this section.

```
<listWords>
 <wordPrior word = 'phone'>
      <priority>10</priority>
      <thema>telephone</thema>
 </wordPrior>
 <wordPrior word = 'contact'>
      <priority>10</priority>
      <thema>telephone</thema>
 </wordPrior>
 <wordPrior word = 'are you from'>
      <priority>4</priority>
      <thema>city</thema>
 </wordPrior>
 <wordPrior word = 'you come from'>
      <priority>6</priority>
      <thema>city</thema>
 </wordPrior>
 <wordPrior word = 'do you live'>
      <priority>4</priority>
      <thema>city</thema>
 </wordPrior>
 <wordPrior word = 'help you'>
      <priority>7</priority>
      <thema>help resp</thema>
 </wordPrior>
 <wordPrior word = 'can i do'>
      <priority>7</priority>
      <thema>help resp</thema>
 </wordPrior>
 <wordPrior word = 'do you want'>
      <priority>4</priority>
      <thema>help resp</thema>
 </wordPrior>
</listWords>
```

Short example from Listwords.xml

```
<ListAnswerOnt>
      <focus thema = 'weather'>
            <genus name = 'sunny'>
                  <markupPhrases markup = '22'>
                        <sentence>The sun will shine #INFO1#. I'll
have to take my hat</sentence>
                  </markupPhrases>
                  <markupPhrases markup = '27'>
                        <sentence>What a pity, it'll be sunny
#INFO1# and I've a lot of work to do</sentence>
                  </markupPhrases>
                  <markupPhrases markup = '72'>
                        <sentence>Yeah, there 'll be a lot of sun
#INFO1#.</sentence>
                  </markupPhrases>
```

```
                    <markupPhrases markup = '77'>
                          <sentence>I'll probably go wandering
#INFO1#, it'll be sunny.</sentence>
                    </markupPhrases>
              </genus>
              <genus name = 'clouds'>
                    <markupPhrases markup = '22'>
                          <sentence>Clouds, clouds, always clouds...
I'm fed up with this weather!</sentence>
                    </markupPhrases>
                    <markupPhrases markup = '27'>
                          <sentence>I really don't like this weather
where we cannot say if there 'll be rain or not</sentence>
                    </markupPhrases>
                    <markupPhrases markup = '72'>
                          <sentence>The sky will be cloudy #INFO1#,
I hope there 'll be no rain</sentence>
                    </markupPhrases>
                    <markupPhrases markup = '77'>
                          <sentence>The sky 'll be cloudy #INFO1#,
we'll probably have some shower</sentence>
                    </markupPhrases>
              </genus>
              </genus>
        </focus>
</ListAnswerOnt>
```

Short example from ListAnswerOntologies.xml

```
<ListAnsWithoutOnt>
 <focus thema = 'exit' action = 'F_Yes'>
       <markupPhrases markup = '22'>
             <output pat = ''>[Huh, y|Y]ou are { not}
[leaving|going] so soon?</output>
             <output pat = ''>Come on! Don't let me alone!</output>
       </markupPhrases>
       <markupPhrases markup = '27'>
             <output pat = ''>Why don't you stay with me?
[;o)|:)|;-)|:.)]</output>
       </markupPhrases>
       <markupPhrases markup = '72'>
             <output pat = ''>[Are you|You are|You're] leaving me{
{right }now}?</output>
             <output pat = ''>See you later mate [;o)|:)|;-
)|:.)]</output>
       </markupPhrases>
       <markupPhrases markup = '77'>
             <output pat = ''>[Do y|Y]ou {really }wish to leave?
[;o)|:)|;-)|:.)]</output>
             <output pat = ''>I nearly forgot my appointement
{again}, I have to go {right now}, bye</output>
       </markupPhrases>
 </focus>
 <focus thema = 'identity' action = 'F_Yes'>
       <markupPhrases markup = '22'>
             <output pat = ''>[You can|Just] call me #NAME#, and
you?</output>
       </markupPhrases>
       <markupPhrases markup = '27'>
```

```
            <output pat = ''>If you want, you can call me #NAME#,
what about your name?</output>
      </markupPhrases>
      <markupPhrases markup = '72'>
            <output pat = ''>My name is #NAME#, and what is [your
name|yours]?</output>
      </markupPhrases>
      <markupPhrases markup = '77'>
            <output pat = ''>My name is #NAME#, and you, what[
is|'s] your name?</output>
      </markupPhrases>
 </focus>
</ListAnsWithoutOnt>
```

Short example from ListAnsWithoutOnt.xml

References

[1]AgentLand, What's a chatterbot?, http://www.agentland.com/pages/learn/chatterbox_challenge/chatterbot1.html, (Date of access 26/05/04).

[2] Rizzo P., Miceli M., Cesta A., Goal-based personalities in lifelike agents realized with hybrid planning techniques, Institute of Psychology of the National Research Council (IP-CNR), Roma, Italy.

[Allport, 1927] Allport G. W., Concepts of Trait and Personality. In: *Psychological Bulletin,* 24, 284-293, 1927.

[Atkinson and al., 1983] Atkinson R. L., Atkinson R. C., Hilgard E. R., Introduction to psychology, Harcourt Brace Jovanovich, Inc, 1983.

[Bartneck C., 2002] Bartneck C., Integrating the OCC Model of Emotions in Embodied Characters, Department of Industrial Design, Eindhoven University of Eindhoven, 2002.

[Bates and al., 1992] Bates J., Loyall A., Reilly W. S., An Architecture for Action, Emotion, and Social Behaviour, Technical report *CMU-CS-92-144*, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1992.

[Bates, 1994] Bates, J., The role of emotion in believable agents, In: *Communications of the ACM*, 37, 7, 123 -125.

[Carbonell, 1980] Carbonell J., Towards a Process Model of Human Personality Traits, *Artificial Intelligence*, 15, 49-74, 1980.

[Debenham, 2003] Debenham J.K., An eNegotiation Framework, In : *Proceedings International Conference AI'03*, Cambridge, UK, 2003.

[Doyle, 2001] Doyle P., The Virtual Theater Project: Playbill, http://www-ksl.stanford.edu/projects/cait/playbill.html, (last revised May 7, 2001) (Date of access 20/05/04).

[Egges and al, 2003] Egges A., Kshirsagar S., Magnenat-Thalmann N., A model for personality and emotion simulation, MIRALab, University of Geneva, 2003.

[Elliot, 1992] Elliott C. D., The Affective Reasoner: A Process model of emotions in a multi-agent system, Unpublished Ph.D. thesis, *The Institute for the Learning Sciences*, Northwestern University, Evanston, Illinois, 1992.

[Esteva and al, 2002] Esteva M., de la Cruz D., Sierra C., ISLANDER : an electronic institutions editor. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy.

[Fiske and Taylor, 1991, p.177] Fiske S. T., Taylor S. E., *Social cognition,* McGraw-Hill, New York, 1991.

[Floridi and Sanders, 2001] Floridi L., Sanders J.W., On the Morality of Artificial Agents, *Computer Ethics: Philosophical Enquiries Conference*, 2001. (URL: www.wolfson.ox.ac.uk/~floridi/pdf/maa.pdf)

[Ford, 1992] Ford M. E., Motivating Humans: Goals, Emotions, and Personal Agency Beliefs, Sage, Newbury Park (CA), 1992.

[Franklin and Grasser, 1996] Franklin S., Graesser A., Is it an Agent, or just a Program? : A Taxonomy for Autonomous Agents, In: *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages*, 1996.

[Hayes-Roth, 1995] Hayes-Roth B., An Architecture for Adaptive Intelligent Systems, *Artificial Intelligence: Special Issue on Agents and Interactivity,* 72, 329-365, 1995.

[Hayes-Roth and van Gent, 1996] Hayes-Roth B., van Gent R., Story-Making with Improvisational Puppets and Actors, Technical Report *KSL-96-05,* Knowledge System Laboratory, Stanford University, 1996.

[Howard and Howard, 2000] Howard P. J., Howard J. M., The Big Five Quickstart: An Introduction to the Five-Factor Model of Personality for Human. Available in http://centacs.com/quik-prt.htm. Last consulted in 15 Mars 2004.

[Isbister and Hayes-Roth, 1998] Isbister K., Hayes-Roth B., Social Implications of Using Synthetic Characters: An examination of a Role-Specific Intelligent Agent, Technical Report no. *KSL 98-01*, 1998.

[Johnstone, 1992] Johnstone K., Impro: Improvistion and the Theatre, Routledge, New York, 1992.

[Koda, 1996] Koda T., Agents with Faces: A Study on the Effect of Personification of Software Agents. Unpublished Master Thesis, MIT Media Lab, Cambridge, 1996.

[Kshirsagar and Magnenat-Thalmann, 2002] Kshirsagar S., Magnenat-Thalmann N., A multilayer personality model, In: *Proceedings of 2nd International Symposium on Smart Graphics*, ACM Press 107–115, 2002.

[Lepuschitz, 2000] Lepuschitz G., A definition for Agents, http://www.arcs.ac.at/publik/fulltext/lepuschitzg_dipl/node67.html, (last revised June 13, 2000) (Date of access 18/05/04)

[Loyall and Bates, 1991] Loyall A. B., Bates J., Hap A Reactive, Adaptive Architecture for Agents, *CMU-CS-91-147*, 1991.

[Milgrom, 2004] Milgrom P., Putting Auction Theory to Work, In : *Cambridge University Press*, 2004.

[Moulin and Rousseau, 1996] Moulin B., Rousseau D., Chapter 9, An approach for modelling and simulating conversations, In: *Essays in Speech Act Theory*, Vanderveken, Daniel and Susumu Kubo (eds.), 175-205, 1996.

[Nathan and Garner, 1997] Nathan P. X., Garner R. G., The Evolution of Intelligent Agents on the Web, http://scorpius.spaceports.com/~robitron/Fiction/paper1_5.html, (last revised, 1997) (Date of access 20/05/04).

[Ortony and al, 1988] Ortony A., Clore G., Collins A., The Cognitive Structure of Emotions, Cambridge: Cambridge University Press, 1988.

[Ortony, 2003] Ortony A., On making believable emotional agents believable, In: *Emotions in humans and artefacts*, R. P. Trapple P. (Ed.), Cambridge: MIT Press, 2003

[Pervin, 1989] Pervin L. A., Goal Concepts in Personality and Social Psychology, Lawrence Erlbaum Associates, Hillsdale (NJ), 1989.

[Reilly and Bates, 1992] Reilly W. S., Bates J., Building emotional agents, Technical Report *CMU-CS-92-143*, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1992.

[Reilly, 1995] Reilly W. S., Bates J., Natural Negotiation for Believable Agents, Technical Report *CMU-CS-95-164,* Carnegie Mellon University, 1995.

[Reilly, 1996] Reilly W. S., Believable Social and Emotional Agents, Ph.D. Dissertation, *CMU-CS-96-138*, May 1996.

[Rizzo and al. 1997] Rizzo P., Veloso M., Miceli M., Cesta A., Personality-Driven Social Behaviors in Believable Agents, In: *proceedings of the AAAI 1997 Fall Symposium on "Socially Intelligent Agents"*, Massachusetts,1997.

[Rizzo, 1998] Rizzo P., Personalities in believable agents: A goal-based model and its realization with an integrated planning architecture, PhD thesis, Centro di Scienza Cognitiva, University di Torino, Torino, 1998.

[Rousseau and Hayes-Roth 1996] Rousseau D., Hayes-Roth B., Personality in Synthetic Agents, Technical Report *KSL.96-21,* Knowledge System Laboratory, Stanford University, 1996.

[Rousseau and Hayes-Roth, 1997] Rousseau D., Hayes-Roth B., A Social-Psychological Model for Synthetic Actors, Knowledge Systems Laboratory Report *KSL 97-07*, Department of Computer Science, Stanford University, 1997.

[Russell and Norvig, 1995] Russell S., Norvig P., *Artificial Intelligence: a Modern Approach*, Englewood Cliffs: Prentice-Hall, Inc, 1995.

[Russell and Norvig, 2003] Russell S. J., Norvig P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2nd edition, 2003.

[Silva and al, 2001] Silva D. R., Siebra C. A., Valadares J. L., Almeida A. L., Frery A. C., da Rocha Falcão J., Ramalho G. L., Synthetic Actor Model for Long-Term Computer Games, 2001.

[Simoff, 2002] Simoff S. J., Informed traders in electronic markets - mining 'living' data to provide context information to a negotiation process, In: *Proceedings of the 3rd International We-B Conference*, 28-29 November, Perth, Australia, 2002.

[Smith and al., 1994] Smith D. C., Cypher A., Spohrer J., KidSim: Programming Agents Without a Programming Language, *Communications of the ACM*, 37, 7, 55-67.

[Sweet, 1978] Sweet J., *Something Wonderful Right Away*, New York: Proscenium Publishers, Inc, 1978.

[Thomas, 1981] Thomas F., Johnston O., *Disney Animation: The Illusion of Life.* Abbeville Press, New York, 1981.

[Trappl, 1997] Trappl, R., *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents,* Paolo Petta, Austrian Research Institute for Artificial Intelligence, Vienna, Austria, EU, Springer LNAI 1195 State-of-the-Art Survey, 1997.

[Walsh, 1998] Walsh N., A Technical Introduction to XML, http://www.xml.com, (last revised October 03, 1998), Date of access 26/05/04)

[Wooldridge and Jennings, 1995] Wooldridge M., Jennings N. R., Agent Theories, Architectures, and Languages: a Survey, in Wooldridge and Jennings Eds., *Intelligent Agents*, Berlin: Springer-Verlag, 1-22, 1995.